PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Cz, 14 maj 2015 08:42:44

Spis treści

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:

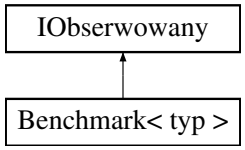
4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy Benchmark< typ >

Modeluje pojęcie Benchmarku.

`#include <Benchmark.hh>`

Diagram dziedziczenia dla Benchmark< typ >



Metody publiczne

- Benchmark (const unsigned int ileProb, unsigned int *const ileDanych, const unsigned int ilePowtorzen)

  Konstruktor 2 argumentowy.

- void Test (Framework ∗I, std::string const nazwaPlikuDane)

  Testowanie algorytmu.
- void DodajObserwatora (IObserwator ∗nowyObserwator)

  Dodaje Obserwatora.
- void UsunObserwatora (IObserwator ∗obserwator)

  Usuwa Obserwatora.
- void PowiadomObserwatorow ()

  Powiadamia Obserwatorów.

Atrybuty prywatne

- unsigned int IleProb

  Ilość prób.
- unsigned int ∗ IleDanych

  Tablica liczności serii.
- unsigned int IlePowtorzen

  Ilość powtórzeń
- std::list< IObserwator ∗ > ListaObserwatorow

  Lista Obserwatorow.

4.1.1 Opis szczegółowy

template<class typ>class Benchmark< typ >

Modeluje pojęcie Benchmarku czyli objektu mierzącego czas wykonywania algoytmu

Definicja w linii 26 pliku Benchmark.hh.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 template<class typ> Benchmark< typ >::Benchmark (const unsigned int *ileProb,* unsigned int ∗const *ileDanych,* const unsigned int *ilePowtorzen*) [inline]

Tworzy objekt klasy Benchmark i inicjuje nową statystykę dla objektu

Parametry

in	*ileProb*	- ilość prób, które zostaną wykonane
in	*ileDanych*	- wkaźnik na tablice z licznościami kolejnych serii
in	*ilePowtorzen*	- ilość powtórzeń każdej serii

Definicja w linii 71 pliku Benchmark.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 template<class typ> void Benchmark< typ >::DodajObserwatora (IObserwator ∗ *nowyObserwator*) [inline], [virtual]

Dodaje obserwatora do listy obserwatorów danego objektu

Parametry

in	*nowyObserwator*	- wskaźnik na objekt będący obserwatorem

Implementuje IObserwowany.

Definicja w linii 113 pliku Benchmark.hh.

4.1.3.2 template<class typ> void Benchmark< typ >::PowiadomObserwatorow ( ) [inline], [virtual]

Wywołuje u wszystkich aktywnych obserwatorów metodę Aktualizuj.

Implementuje IObserwowany.

Definicja w linii 133 pliku Benchmark.hh.

4.1.3.3 template<class typ> void Benchmark< typ >::Test (Framework * *I*, std::string const *nazwaPlikuDane*) [inline]

Metoda testuje algorytm w okreslonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego pez użytkownika

Parametry

in	*I*	- objekt klasy na której zostanie przeprowadzony test
in	*nazwaPlikuDane*	- nazwa pliku z danymi do wczytania

Definicja w linii 87 pliku Benchmark.hh.

4.1.3.4 template<class typ> void Benchmark< typ >::UsunObserwatora (IObserwator * *obserwator*) [inline], [virtual]

Usuwa danego obserwatora z listy obserwatorów

Parametry

in	*obserwator*	- wskaźnik na obserwatora który ma zostać usunięty

Implementuje IObserwowany.

Definicja w linii 124 pliku Benchmark.hh.

4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 template<class typ> unsigned int* Benchmark< typ >::IleDanych [private]

Tablica z licznościami elementów dla kojenych serii

Definicja w linii 42 pliku Benchmark.hh.

4.1.4.2 template<class typ> unsigned int Benchmark< typ >::IlePowtorzen [private]

Ilość powtórzeń każdej serii

Definicja w linii 50 pliku Benchmark.hh.

4.1.4.3 template<class typ> unsigned int Benchmark< typ >::IleProb [private]

Ilość powtórzeń każdej seriii

Definicja w linii 34 pliku Benchmark.hh.

4.1.4.4 template<class typ> std::list<IObserwator*> Benchmark< typ >::ListaObserwatorow [private]

Lista aktywnych obserwatorów danego objektu

Definicja w linii 57 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

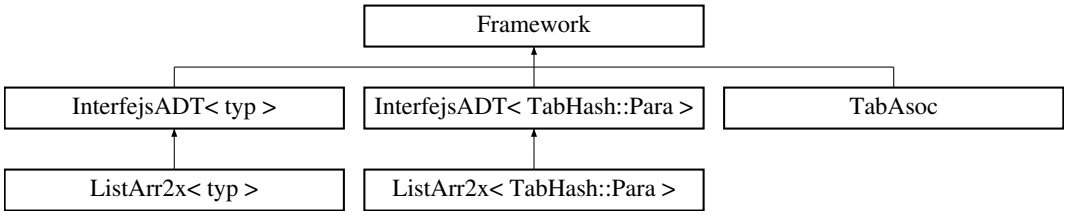
- Benchmark.hh

4.2 Dokumentacja klasy Framework

Modeluje interfejs programu.

`#include <Framework.hh>`

Diagram dziedziczenia dla Framework



Metody publiczne

- virtual void WczytajDane (const char ∗nazwaPliku, const unsigned int n)=0

  Wczytanie danych z pliku.
- virtual void Start (std::fstream &plik, const unsigned int k)=0

  Wykonanie części obliczeniowej programu.
- virtual void Zwolnij ()=0

  Zwalnia pamięć po teście.

4.2.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanch w ramach kursu.

Definicja w linii 25 pliku Framework.hh.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 virtual void Framework::Start (std::fstream & *plik*, const unsigned int *k*) [pure virtual]

Metoda w której implementowana jest część obliczeniowa programu, której czas wykonania zostanie zmierzony.

Parametry

in	*k*	- ilość elementów dla których mają zostać wykonane obliczenia.
in	*plik*	- plik z którego wczytujemy dane

Implementowany w ListArr2x< typ >, ListArr2x< TabHash::Para >, InterfejsADT< typ >, InterfejsADT< TabHash::Para > i TabAsoc.

4.2.2.2 virtual void Framework::WczytajDane (const char ∗ *nazwaPliku*, const unsigned int *n*) [pure virtual]

Wczytuje zadaną ilość danych do przetworzenia z pliku o zadanej nazwie.

Parametry

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania

Implementowany w ListArr2x< typ >, ListArr2x< TabHash::Para >, InterfejsADT< typ >, InterfejsADT< TabHash::Para > i TabAsoc.

4.2.2.3 virtual void Framework::Zwolnij ( ) [pure virtual]

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

Implementowany w ListArr2x< typ >, ListArr2x< TabHash::Para >, InterfejsADT< typ >, InterfejsADT< TabHash::Para > i TabAsoc.

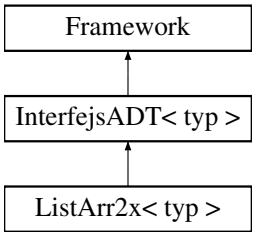
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Framework.hh

4.3 Dokumentacja szablonu klasy InterfejsADT< typ >

`#include <InterfejsADT.hh>`

Diagram dziedziczenia dla InterfejsADT< typ >



Metody publiczne

- virtual void push (const typ dana, const unsigned int pole)=0
  Dodaje kolejny element.
- virtual void pop (const unsigned int pole)=0
  Pobiera element.
- virtual unsigned int size () const =0
  Liczność elemetów.
- void WczytajDane (const char *nazwaPliku, const unsigned int n)=0
  Wczytanie danych z pliku.
- void Start (std::fstream &plik, const unsigned int k)=0
  Wykonanie części obliczeniowej programu.
- virtual void Zwolnij ()=0
  Zwalnia pamięć

4.3.1 Opis szczegółowy

template<class typ>class InterfejsADT< typ >

\ brief Definiuje interfejs użytkownika

Definiuje interfejs użytkownika dla listy, stosu i kolejki.

Definicja w linii 13 pliku InterfejsADT.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 template<class typ> virtual void InterfejsADT< typ >::pop (const unsigned int *pole*) [pure virtual]

Pobiera element z typu danych

Parametry

in	*pole*	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola z ktore pobiera element

Zwracane wartości

zwraca	wartość danego elementu

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

4.3.2.2 template<class typ> virtual void InterfejsADT< typ >::push (const typ *dana,* const unsigned int *pole*) [pure virtual]

Dodaje kolejny element do typu danych

Parametry

in	*dana*	- element który chcemy dorzucić do naszego typu
in	*pole*	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola na które chcemy dodać element

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

4.3.2.3 template<class typ> virtual unsigned int InterfejsADT< typ >::size ( ) const [pure virtual]

Informuje o liczności elementów obecnie przechowywanych

Zwracane wartości

zwraca	ilość przechowywanych elementów

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

4.3.2.4 template<class typ> void InterfejsADT< typ >::Start (std::fstream & *plik,* const unsigned int *k*) [pure virtual]

Metoda w której implementowana jest część obliczeniowa programu, której czas wykonania zostanie zmierzony.

Parametry

in	*k*	- ilość elementów dla których mają zostać wykonane obliczenia.
in	*plik*	- plik z którego wczytujemy dane

Implementuje Framework.

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

4.3.2.5 template<class typ> void InterfejsADT< typ >::WczytajDane (const char * *nazwaPliku,* const unsigned int *n*) [pure virtual]

Wczytuje zadaną ilość danych do przetworzenia z pliku o zadanej nazwie.

Parametry

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania

Implementuje Framework.

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

4.3.2.6 template<class typ> virtual void InterfejsADT< typ >::Zwolnij ( ) [pure virtual]

Zwalnia pamięć zajmowaną przez daną strukturę

Implementuje Framework.

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

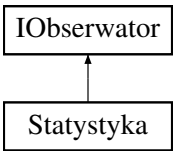
- InterfejsADT.hh

4.4 Dokumentacja klasy IObserwator

Klasa IObserwator.

`#include <IObserwator.hh>`

Diagram dziedziczenia dla IObserwator



Metody publiczne

- virtual void Aktualizuj ()=0
  Aktualizuj.

4.4.1 Opis szczegółowy

Plik zawira definicję klasy IObsereator.

The IObserwator class

Klasa modeluje interfejs objektu będącego obserwatorem.

Definicja w linii 15 pliku IObserwator.hh.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 virtual void IObserwator::Aktualizuj ( ) [pure virtual]

Aktualizuje dane na podstawie wydarzenie w objekcie obserowanym.

Implementowany w Statystyka.

Dokumentacja dla tej klasy została wygenerowana z pliku:

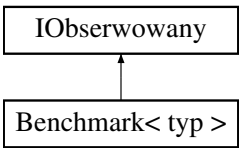
- IObserwator.hh

4.5 Dokumentacja klasy IObserwowany

The IObserwowany class.

`#include <IObserwowany.hh>`

Diagram dziedziczenia dla IObserwowany



Metody publiczne

- virtual void DodajObserwatora (IObserwator ∗nowyObserwator)=0

  Dodaje Obserwatora.
- virtual void UsunObserwatora (IObserwator ∗obserwator)=0

  Usuwa Obserwatora.
- virtual void PowiadomObserwatorow ()=0

  Powiadamia Obserwatorów.

4.5.1 Opis szczegółowy

Klasa czysto wirtualna modelująca interfejs objektu obserwowanego.

Definicja w linii 17 pliku IObserwowany.hh.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 virtual void IObserwowany::DodajObserwatora (IObserwator ∗ *nowyObserwator*) [pure virtual]

Dodaje nowego obserwatora do listy oserwatorów danego objektu.

Parametry

in	*nowyObserwator*	- wkaźnik na dodawanego obserwatora

Implementowany w Benchmark< typ >.

4.5.2.2 virtual void IObserwowany::PowiadomObserwatorow ( ) [pure virtual]

Powiadamia obseratorów o wydarzeniu.

Implementowany w Benchmark< typ >.

4.5.2.3 virtual void IObserwowany::UsunObserwatora (IObserwator ∗ *obserwator*) [pure virtual]

Usuwa danego obserwatora z listy obserwatorów danego objektu.

Parametry

in	*obserwator*	- obserwator do usunięcia z listy

Implementowany w Benchmark< typ >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

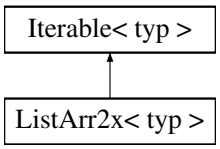
- IObserwowany.hh

4.6 Dokumentacja szablonu klasy Iterable< typ >

Definicja Iterable.

```
#include <Iterable.hh>
```

Diagram dziedziczenia dla Iterable< typ >



Metody publiczne

- virtual typ operator[] (unsigned int i)=0

  operator []
- virtual typ & RefEnd ()=0

  RefEnd.

4.6.1 Opis szczegółowy

template<class typ>class Iterable< typ >

Plik zawiera definicje interfejsu Iterable

The Iterable class

Klasa modeluje interfejs umożliwiajacy przeglądanie kontenera oraz uzystkiwanie referencji do jego ostatniego pola co jest wymagane w obecnej implementacji tablicy asocjacyjnej

Definicja w linii 21 pliku Iterable.hh.

4.6.2 Dokumentacja funkcji składowych

4.6.2.1 template<class typ> virtual typ Iterable< typ >::operator[] (unsigned int *i*) [pure virtual]

Przeciążenie operatora [] w celu umożliwienia przeglądania kontenera

Parametry

in	*i*	- indeks elementu

Zwracane wartości

-	zwraca wartośc znajdującą się na danym indeksie

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

4.6.2.2 template<class typ> virtual typ& Iterable< typ >::RefEnd ( ) [pure virtual]

Zwraca referencję do ostatniego elementu kontenera umożliwiając przypisanie tam nowego elementu.

Zwracane wartości

-	referencja do ostatniego pola listy

Implementowany w ListArr2x< typ > i ListArr2x< TabHash::Para >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

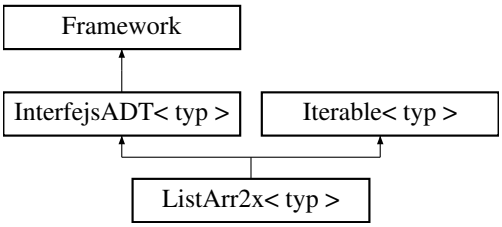
- Iterable.hh

4.7 Dokumentacja szablonu klasy ListArr2x< typ >

Modeluje pojęcie Listy (array)

`#include <ListArr2x.hh>`

Diagram dziedziczenia dla ListArr2x< typ >



Metody publiczne

- ListArr2x ()
  Konstruktor bezarumentowy.
- void push (const typ dana, const unsigned int pole)
  Dodaje element do ListyArr2x.
- void pop (const unsigned int pole)
  Pobiera element z ListyArr2x.
- unsigned int size () const
  Wielkość listy.
- void Start (std::fstream &plik, const unsigned int k)
  Metoda której czas wykonania jest testowany.
- void WczytajDane (const char ∗nazwaPliku, unsigned int n)
  Wczytuje dane z pliku.
- void Zwolnij ()
  Zwalnia pamięć
- typ operator[] (unsigned int i)
  operator []
- typ & RefEnd ()
  RefEnd.

Metody prywatne

- void UsunZListy (const unsigned int pole)
  UsunZListy.
- void DodajDoListy (const typ dana, const unsigned int pole)
  DodajDoListy.

Atrybuty prywatne

- typ ∗ tab
  Wkaźnik na dynamiczną tablicę
- unsigned int RozmiarT
  Rozmiar tablicy.
- unsigned int RozmiarL
  Rozmiar Listy.

4.7.1 Opis szczegółowy

template<class typ>class ListArr2x< typ >

Modeluje pojęcie Listy opartej na dynamicznej tablicy. Dodając elementy zwiększa tablicę dwukrotnie, jeżeli brakuje miejsca. a

Definicja w linii 19 pliku ListArr2x.hh.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 template<class typ> ListArr2x< typ >::ListArr2x ( ) [inline]

Kontruktor alokujący tablicę jednoelementową z której będzie tworzona lista

Definicja w linii 87 pliku ListArr2x.hh.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 template<class typ> void ListArr2x< typ >::DodajDoListy (const typ *dana,* const unsigned int *pole*) [inline], [private]

Dodaje daną do listy na określony indeks

Parametry

dana	- wartość która ma zostać umieszczona na liście
pole	- indeks pola na którym ma zostać umieszczona wartość

Definicja w linii 67 pliku ListArr2x.hh.

4.7.3.2 template<class typ> typ ListArr2x< typ >::operator[] (unsigned int *i*) [inline], [virtual]

Przeciążenie operatora [] w celu umożliwienia przeglądania listy

Parametry

in	*i*	- indeks elementu

Zwracane wartości

-	zwraca wartośc znajdującą się na danym indeksie

Implementuje Iterable< typ >.

Definicja w linii 223 pliku ListArr2x.hh.

4.7.3.3 template<class typ> void ListArr2x< typ >::pop (const unsigned int *pole*) [inline], [virtual]

Pobiera element z ListyArr2x usuwając go z niej i zmniejszając rozmiar o połowę w przypadku przekroczenia stosunku 1:4 (RozmiarL:RozmiarT)

param[in] - pole - nr pola z którgo chcemy pobrać element (indeksowane od 0)

Implementuje InterfejsADT< typ >.

Definicja w linii 138 pliku ListArr2x.hh.

4.7.3.4 template<class typ> void ListArr2x< typ >::push (const typ *dana,* const unsigned int *pole*) [inline], [virtual]

Dodaje nowy element do ListyArr2x

Parametry

in	*dana*	- element który chcemy umieścić na liście
in	*pole*	- nr pola na którym chcemy umieścić element jeżeli chcesz umieścić na początku listy podaj wartość 0, na końcu warość size()

Implementuje InterfejsADT< typ >.

Definicja w linii 103 pliku ListArr2x.hh.

4.7.3.5 template<class typ> typ& ListArr2x< typ >::RefEnd ( ) [inline], [virtual]

Zwraca referencję do ostatniego elementu listy umożliwiając przypisanie tam nowego elementu.

Zwracane wartości

-	referencja do ostatniego pola listy

Implementuje Iterable< typ >.

Definicja w linii 235 pliku ListArr2x.hh.

4.7.3.6 template<class typ> unsigned int ListArr2x< typ >::size ( ) const [inline], [virtual]

Informuje o ilości elementów znajdujących się na LiścieArr1

Zwracane wartości

-	zwraca liczbę elementów ListyArr1

Implementuje InterfejsADT< typ >.

Definicja w linii 171 pliku ListArr2x.hh.

4.7.3.7 template<class typ> void ListArr2x< typ >::Start (std::fstream & *plik,* const unsigned int *k*) [inline], [virtual]

Metoda testująca czas wczytania n elementów na ListęArr2x

Parametry

in	*k*	- ilość elementów do wczytania
in	*plik*	- uchwyt to pliku z danymi

Implementuje InterfejsADT< typ >.

Definicja w linii 181 pliku ListArr2x.hh.

4.7.3.8 template<class typ> void ListArr2x< typ >::UsunZListy (const unsigned int *pole*) [inline], [private]

Usuwa z listy element o podanym indeksie

Parametry

in	*pole*	- indeks elementu do usunięcia.

Definicja w linii 49 pliku ListArr2x.hh.

4.7.3.9 template<class typ> void ListArr2x< typ >::WczytajDane (const char * *nazwaPliku,* unsigned int *n*) [inline], [virtual]

Wczytuje dane z pliku do ListArr2x

param[in] nazwaPliku - nazwa pliku z danymi param[in] n - ilość danych do wczytania, 0 oznacza wszystkie dane z pliku

Implementuje InterfejsADT< typ >.

Definicja w linii 194 pliku ListArr2x.hh.

4.7.3.10 template<class typ> void ListArr2x< typ >::Zwolnij ( ) [inline], [virtual]

Zwalnia pamięć zaalokowaną przez ListArr2x

Implementuje InterfejsADT< typ >.

Definicja w linii 210 pliku ListArr2x.hh.

4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 template<class typ> unsigned int ListArr2x< typ >::RozmiarL [private]

Aktualny rozmiar ListyArr2x

Definicja w linii 40 pliku ListArr2x.hh.

4.7.4.2 template<class typ> unsigned int ListArr2x< typ >::RozmiarT [private]

Aktualny rozmiar tablicy.

Definicja w linii 33 pliku ListArr2x.hh.

4.7.4.3 template<class typ> typ* ListArr2x< typ >::tab [private]

Wskaźnik na dynamiczną tablicę tworzącą ListęArr2x

Definicja w linii 26 pliku ListArr2x.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ListArr2x.hh

4.8 Dokumentacja struktury TabHash::Para

Para wartości klucz - wartość

Metody publiczne

- Para (const int wart, const std::string key)
  Konstruktor 2 argumentowy.
- Para (const int i)
  Konstruktor 1 argumentowy.
- Para ()
  Konstruktor bezarumentowy.
- void operator= (const Para p)
  Operator przypisania.

Atrybuty publiczne

- std::string Klucz
  Klucz.
- int Wartosc
  Wartość

4.8.1 Opis szczegółowy

Struktura modeluje nierozłączny element Tablicy Haszującej czyli parę klucz - wartość

Definicja w linii 31 pliku TabHash.hh.

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 TabHash::Para::Para (const int *wart,* const std::string *key*)

Definicja metod Tablicy Haszującej.

Dwuarumentowy onstruktor nierozłącznej Pary (Klucz i Wartosc) Tworzy nowy objekt inicjując go podanymi wartościami

Parametry

in	*wart*	- wartość, którą inicjujemy objekt
in	*key*	- klucz, którym inicjujemy objekt

Plik zawiera definicję metod klasy TabHash

Definicja w linii 11 pliku TabHash.cpp.

4.8.2.2 TabHash::Para::Para (const int *i*)

Jednoargumentowy konstruktor nierozłącznej pary (Warotsc i Klucz) Tworzy nowy objekt inicjując go kluczem: "" i wartością i

Parametry

in	*i*	- wartosc którą zostanie zainicjowany objekt

Definicja w linii 17 pliku TabHash.cpp.

4.8.2.3 TabHash::Para::Para ( )

Bezargumentowy konstruktor nierozłącznej pary (Klucz i Wartość) Tworzy nowy objekt inicjując go kluczem: "" i wartością -1

Definicja w linii 22 pliku TabHash.cpp.

4.8.3 Dokumentacja funkcji składowych

4.8.3.1 void TabHash::Para::operator= (const Para *p*)

Przeciążenie opratora przypisania - kopiuje i przypisuje wartości pól

Parametry

in	*p*	- objekt który chcemy skopiować

Definicja w linii 27 pliku TabHash.cpp.

4.8.4 Dokumentacja atrybutów składowych

4.8.4.1 std::string TabHash::Para::Klucz

Klucz pod którym przechowywana jest wartość

Definicja w linii 39 pliku TabHash.hh.

4.8.4.2 int TabHash::Para::Wartosc

Wartość przechowywana w Tablicy Haszującej pod kluczem

Definicja w linii 47 pliku TabHash.hh.

Dokumentacja dla tej struktury została wygenerowana z plików:

- TabHash.hh

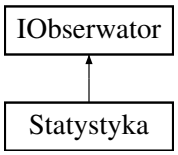
- TabHash.cpp

4.9 Dokumentacja klasy Statystyka

Modeluje pojęcie statystyki.

`#include <Statystyka.hh>`

Diagram dziedziczenia dla Statystyka



Metody publiczne

- Statystyka (const unsigned int iloscProb, unsigned int ∗proby, const unsigned int ilePowtorzen)

  Konstruktor z dwoma pramametrami.
- ∼Statystyka ()

  Destruktor - zwaknia pamięć
- void ZapiszStaty (std::string nazwaPliku) const

  Zapisuje statysykę do pliku.
- void Aktualizuj ()

  Aktualizuj.

Atrybuty prywatne

- unsigned int IleProb

  Ilość prób.
- unsigned int ∗ Proba

  Tablica z rozmiarami prób.
- double ∗ Czas

  Średni czas wykonania danej próby.
- double SumaCzasuProby

  Suma Czasu Proby.
- unsigned int IloscPowtorzen

  Ilość Powtórzeń
- unsigned int LicznikPowtorzen

  Licznik Powtórzeń
- unsigned int LicznikProb

  Licznik Prób.
- Stoper ∗ MojStoper

  Stoper.

4.9.1 Opis szczegółowy

Modeluje pojęcie statystyki, czyli średnich czasów wykonania metody dla różnyuch wielkości prób.

Definicja w linii 27 pliku Statystyka.hh.

4.9.2 Dokumentacja konstruktora i destruktora

4.9.2.1 Statystyka::Statystyka (const unsigned int *iloscProb,* unsigned int * *proby,* const unsigned int *ilePowtorzen*)

Konstruktor z dwoma paramatremi tworzy dynamiczne tablice przechowujące statystykę oraz wypełnia rozmiary prób.

Parametry

in	*iloscProb*	- liczbosc prob w ksperymencie
in	*proby*	- tablica z licznościami prób.
in	*ilePowtorzen*	- ilość powtórzeń każdego rozmiaru próby

Definicja w linii 12 pliku Statystyka.cpp.

4.9.2.2 Statystyka::∼Statystyka ( ) [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablicy przechowujące statystykę.

Definicja w linii 109 pliku Statystyka.hh.

4.9.3 Dokumentacja funkcji składowych

4.9.3.1 void Statystyka::Aktualizuj ( ) [virtual]

Aktualizuje pozyskiwane dane dotyczące wyników testu: Jeżeli stoper nie odlicza to uruchamia odliczanie, Jeżeli stper odlicza to go zatrzymuje i sumuje czasy powtórzeń. Gdy nasąpi wykonanie wszystkich pomiarów w próbie to uzupełnia talicę przechowywujacą średnie czasy każdej próby.

Implementuje IObserwator.

Definicja w linii 44 pliku Statystyka.cpp.

4.9.3.2 void Statystyka::ZapiszStaty (std::string *nazwaPliku*) const

Zapisuje statystystykę do pliku o nazwie podanej w argumencie. Plik zapisany zostaje w sposób, gdzie każda nowa linia wygląda następująco: RozmiarPróby,ŚredniCzas czas wyrażony jest w ms.

Parametry

in	*nazwaPliku*	- nazwa pliku do którego ma zostać zapisanaza statystyka

Definicja w linii 25 pliku Statystyka.cpp.

4.9.4 Dokumentacja atrybutów składowych

4.9.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 51 pliku Statystyka.hh.

4.9.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 35 pliku Statystyka.hh.

4.9.4.3 unsigned int Statystyka::IloscPowtorzen [private]

Przechowuje ilość wykonywanych powtórzeń pojedyńczego testu.

Definicja w linii 65 pliku Statystyka.hh.

4.9.4.4 unsigned int Statystyka::LicznikPowtorzen [private]

Zlicza ilosc wykonanych powtórzeń w danej próbie.

Definicja w linii 72 pliku Statystyka.hh.

4.9.4.5 unsigned int Statystyka::LicznikProb [private]

Zlicza ilosć prób wykonanych prób.

Definicja w linii 79 pliku Statystyka.hh.

4.9.4.6 Stoper* Statystyka::MojStoper [private]

Stoper wykorzystywany do pomiaru czasu.

Definicja w linii 86 pliku Statystyka.hh.

4.9.4.7 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 43 pliku Statystyka.hh.

4.9.4.8 double Statystyka::SumaCzasuProby [private]

Przechowuje sumę czasów pojedyńczych powtórzeń z danej próby.

Definicja w linii 58 pliku Statystyka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Statystyka.hh
- Statystyka.cpp

4.10 Dokumentacja klasy Stoper

Klasa Stoper.

`#include <Stoper.hh>`

Metody publiczne

- Stoper ()
  Stoper.
- void Start ()
  Start.
- void Stop ()
  Stop.
- void Reset ()
  Reset.
- double DajPomiar () const
  Pomiar.
- bool CzyOdmierza () const
  Czy Odmierza.

Atrybuty prywatne

- double CzasPoczatkowy
  Czas Początkowy.
- double CzasKoncowy
  Czas Końcowy.
- bool CzyLiczy
  Czy Liczy.

4.10.1 Opis szczegółowy

Plik zawiera definicję klasy Stoper.

The Stoper class

Klasa modeluje stoper niezbędny do odliczania czasu.

Definicja w linii 18 pliku Stoper.hh.

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 Stoper::Stoper ( )

Kontruktor bezarumentowy zeruje czasy i ustawia wartość pola CzyLiczy na false.

Definicja w linii 3 pliku Stoper.cpp.

4.10.3 Dokumentacja funkcji składowych

4.10.3.1 bool Stoper::CzyOdmierza ( ) const

Informuje czy stoper aktualinie liczy czy nie.

Zwracane wartości

true	- gdy odlicza
false	- gdy nie odlicza

Definicja w linii 29 pliku Stoper.cpp.

4.10.3.2 double Stoper::DajPomiar ( ) const

Wyłuskuje czas pomiaru w ms.

Zwracane wartości

zwrca	czas pomiaru wyrażon w ms

Definicja w linii 25 pliku Stoper.cpp.

4.10.3.3 void Stoper::Reset ( )

Resetuje stoper.

Definicja w linii 19 pliku Stoper.cpp.

4.10.3.4 void Stoper::Start ( )

Uruchamia odliczanie czasu.

Definicja w linii 9 pliku Stoper.cpp.

4.10.3.5 void Stoper::Stop ( )

Zatrzymuje odliczanie czasu.

Definicja w linii 14 pliku Stoper.cpp.

4.10.4 Dokumentacja atrybutów składowych

4.10.4.1 double Stoper::CzasKoncowy [private]

Czas w którym odliczanie czasu zostało zatrzymane.

Definicja w linii 32 pliku Stoper.hh.

4.10.4.2 double Stoper::CzasPoczatkowy [private]

Czas w którym stoper zaczął odliczać.

Definicja w linii 25 pliku Stoper.hh.

4.10.4.3 bool Stoper::CzyLiczy [private]

Zmienna przechowuje wartośc true gdy stoper aktualnie odlicza czas, lub false gdy jest zatrzymany.

Definicja w linii 40 pliku Stoper.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

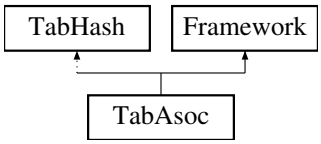
- Stoper.hh
- Stoper.cpp

4.11 Dokumentacja klasy TabAsoc

Definicja klasy TabAsoc.

`#include <TabAsoc.hh>`

Diagram dziedziczenia dla TabAsoc



Metody publiczne

- const int operator() (const std::string klucz) const

  Przeciążenie operatora()
- int & operator[] (const std::string klucz)

  Przeciążenie operatora[].
- void WczytajDane (const char ∗nazwaPliku, const unsigned int n)

  Wczytuje dane.
- void Start (std::fstream &plik, const unsigned int k)

  Obliczenia do pomiarów.
- void Zwolnij ()

  Zwalnia pamięć

Dodatkowe Dziedziczone Składowe

4.11.1 Opis szczegółowy

Plik zawiera definicję klasy TabAsoc

Modeluje Tablice Asocjacyjną

Klasa TabAsoc modeluje pojęcie Tablicy Asocjacyjnej zaimplementowanej jako Tablica Haszująca

Definicja w linii 21 pliku TabAsoc.hh.

4.11.2 Dokumentacja funkcji składowych

4.11.2.1 const int TabAsoc::operator() (const std::string *klucz*) const

Definicje metod TabAsoc.

Przeciążenie operatora() w celu umożliwiena odczytu wartości z tablicy za pomocą klucza

Parametry

in	*klucz*	- klucz pod jakim chcemy znaleźć wartość

Zwracane wartości

-	zwraca wartość znajdującą sie pod danym kluczem, lub -1 w przypadku gdy nie znaleziono pasującego klucza w tablicy

Plik zawiera fedinicje metod Tablicy Asocjacyjnej

Definicja w linii 10 pliku TabAsoc.cpp.

4.11.2.2 int & TabAsoc::operator[] (const std::string *klucz*)

Przeciążenie operatora[] w celi umożliwienia zapisania nowej wartości do tablicy pod wskazanym kluczem

Parametry

in	*klucz*	- klucz pod którym chcemy zapisać daną

Zwracane wartości

-	zwraca referencje do miejsca przechowywanai danej

Definicja w linii 14 pliku TabAsoc.cpp.

4.11.2.3 void TabAsoc::Start (std::fstream & *plik,* const unsigned int *k*) [virtual]

Metoda niezbędna do wykonania Benchmarka w celu zmierzenia czasu zapisu i odczytu z tablicy dancyh

Parametry

in	*plik*	- referencja do owtartego pliku z danymi
in	*k*	- ilość elementów na których zostanie przeprowadzony test

Implementuje Framework.

Definicja w linii 46 pliku TabAsoc.cpp.

4.11.2.4 void TabAsoc::WczytajDane (const char * *nazwaPliku,* const unsigned int *n*) [virtual]

Wcztuje dane do Tablicy Haszującej z pliku, w kótym linijka po linijce są podane kolejne wartości klucz, wartość.

Parametry

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania

Implementuje Framework.

Definicja w linii 22 pliku TabAsoc.cpp.

4.11.2.5 void TabAsoc::Zwolnij ( ) [virtual]

Zwalnia pamięć pomiędzy kolejnymi seriami testów - czyści tylko przechowywane wartości

Implementuje Framework.

Definicja w linii 18 pliku TabAsoc.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

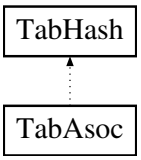
- TabAsoc.hh
- TabAsoc.cpp

4.12 Dokumentacja klasy TabHash

Tablica Haszująca.

`#include <TabHash.hh>`

Diagram dziedziczenia dla TabHash



Komponenty

- struct Para

  Para wartości klucz - wartość

Metody chronione

- const int Pobierz (const std::string szukanyKlucz) const

  Pobiera wartość z Tablicy.
- int & Dodaj (const std::string nowyKlucz)

  Dodaje elemet do tablicy.
- TabHash ()

  Konstruktor bezargumentowy.
- ∼TabHash ()

  Destruktor.

Metody prywatne

- const int DajZListy (const unsigned int pozycja, const std::string szukanyKlucz) const

  Szuka wartości pod kluczemklucz.
- unsigned int H (const std::string klucz) const

  Funkcja haszująca.

Atrybuty prywatne

- ListArr2x< Para > * _Tab [ROZMIAR]

  Lista Par.

4.12.1 Opis szczegółowy

Klasa modeluje pojęcie Tablicy Haszującej

Definicja w linii 22 pliku TabHash.hh.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 TabHash::TabHash ( ) [protected]

Konstruktor bezarumentowy inicjuje tablicę pustymi listami

Definicja w linii 67 pliku TabHash.cpp.

4.12.2.2 TabHash::∼TabHash ( ) [protected]

Destruktor - zwalnia pamięć po listach znajcujących się w tablicy.

Definicja w linii 72 pliku TabHash.cpp.

4.12.3 Dokumentacja funkcji składowych

4.12.3.1 const int TabHash::DajZListy (const unsigned int *pozycja,* const std::string *szukanyKlucz*) const [private]

Przeszukuje Listę znajdującą się na podanej pozycji Tablicy Haszującej w celu znalezienia pasującego klucza

Parametry

in	*pozycja*	- pozycja Tablicy Haszującej na której znajduje się Lista do przeszukania
in	*szukanyKlucz*	- klucz który ma zostać znaleziony

Zwracane wartości

-	zwraca wartość przechowywaną pod danym kluczem

Definicja w linii 33 pliku TabHash.cpp.

4.12.3.2 int & TabHash::Dodaj (const std::string *nowyKlucz*) [protected]

Dodaje element (daną oraz jej klucz) do Tablicy Haszującej

Parametry

in	*nowyKlucz*	- klucz pod którym przechowujemy daną

Definicja w linii 60 pliku TabHash.cpp.

4.12.3.3 unsigned int TabHash::H (const std::string *klucz*) const [private]

Funkcja sumuje wartości liczbowe kodu ASCII liter klucza i na ich podstawie generuje numer indeksu

Parametry

in	*klucz*	- klucz do haszowania

Zwracane wartości

-	zwraca numer indeksu Tablicy Haszującej

Definicja w linii 46 pliku TabHash.cpp.

4.12.3.4 const int TabHash::Pobierz (const std::string *szukanyKlucz*) const [protected]

Pobiera wartość przechowywaną pod zadanycm kluczem z Tablicy Haszuącej

Parametry

in	*szukanyKlucz*	- klucz pod którym szukamy wartości

Zwracane wartości

-	zwraca wartość przechowywaną pod kluczem

Definicja w linii 55 pliku TabHash.cpp.

4.12.4 Dokumentacja atrybutów składowych

4.12.4.1 ListArr2x<Para>* TabHash::_Tab[ROZMIAR] [private]

Lista przechowująca pary: wartość - klucz o takim samym hashu

Definicja w linii 98 pliku TabHash.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- TabHash.hh
- TabHash.cpp

5 Dokumentacja plików

5.1 Dokumentacja pliku Benchmark.hh

Definicja klasy Benchmark.

```
#include "Framework.hh"
#include <ctime>
#include "Statystyka.hh"
#include "IObserwowany.hh"
#include <list>
```

Komponenty

- class Benchmark< typ >

  Modeluje pojęcie Benchmarku.

5.1.1 Opis szczegółowy

Plik zawiera definicję klasy Benchmark wraz z definicją jej metod.

Definicja w pliku Benchmark.hh.

5.2 Dokumentacja pliku Framework.hh

Definicja klasy Framework.

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

Komponenty

- class Framework

  Modeluje interfejs programu.

5.2.1 Opis szczegółowy

Plik zawiera definicję abstrakcyjnej klasy Framework, która tworzy interfejs dla programów implementowanych podczas zajęć laboratoryjnych z PAMSI.

Definicja w pliku Framework.hh.

5.3 Dokumentacja pliku InterfejsADT.hh

```
#include "Framework.hh"
```

Komponenty

- class InterfejsADT< typ >

5.4 Dokumentacja pliku IObserwator.hh

Komponenty

- class IObserwator

  Klasa IObserwator.

5.5 Dokumentacja pliku IObserwowany.hh

```
#include "IObserwator.hh"
```

Komponenty

- class IObserwowany

  The IObserwowany class.

Definicje

- #define IOBSERWOWANY_HH

  Interfejs obserwowanego.

5.5.1 Dokumentacja definicji

5.5.1.1 #define IOBSERWOWANY_HH

W pliku zawarta jest definicja interfejsu obserwowanego

Definicja w linii 8 pliku IObserwowany.hh.

5.6 Dokumentacja pliku Iterable.hh

```
#include <iostream>
```

Komponenty

- class Iterable< typ >

  Definicja Iterable.

5.7 Dokumentacja pliku ListArr2x.hh

Definicja klasy ListArr1.

```
#include "InterfejsADT.hh"
#include "Iterable.hh"
```

Komponenty

- class ListArr2x< typ >

  Modeluje pojęcie Listy (array)

5.7.1 Opis szczegółowy

Plik zawiera definicję klasy ListaArr2x ujętej w szablon typu wraz z jej składowymi metofdami.

Definicja w pliku ListArr2x.hh.

5.8 Dokumentacja pliku main.cpp

Moduł główny programu.

```
#include "../inc/TabAsoc.hh"
#include "../inc/Statystyka.hh"
#include "../inc/Benchmark.hh"
```

Definicje

- #define ILOSC_POWTORZEN 10

  Ilośc powtórzeń danej próby.
- #define ILOSC_PROB 8

  Ilość prób.

Funkcje

- int main (int argc, char ∗argv[])

5.8.1 Opis szczegółowy

Program wkonuje serię 10 pomiarów czasu wykonania metody start (badanie czasu zapisu i odczytu do/z Tablicy Asocjacyjnej dla różncyh wielkości problemu obliczeniowego. Jako plik wynikowy otrzymujemy plik z czasami poświęconymi przez program na zapis/odczyt n danych z tablicy.

WYMAGANIA: Plik z danymi musi być w formacie takim, że każda linia to kolejno "klucz wartość"

Klucze muszą być sześcioznakowymi ciągami stringów składających się wyłącznie z małych liter.

Wartości mogą być dowolnym intem

OBSŁUGA PROGRAMU: Aby wywołać program należy w lini poleceń wywołać jego nazę np: "./a.out"

Definicja w pliku main.cpp.

5.8.2 Dokumentacja definicji

5.8.2.1 #define ILOSC_POWTORZEN 10

Ilośc powtórzeń danej próby

Definicja w linii 39 pliku main.cpp.

5.8.2.2 #define ILOSC_PROB 8

Ilość prób = ilość rozmiarów prób

Definicja w linii 47 pliku main.cpp.

5.8.3 Dokumentacja funkcji

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

Definicja w linii 49 pliku main.cpp.

5.9 Dokumentacja pliku Pliki.cpp

Definicje funkcji obslugi plikow.

```
#include "../inc/Pliki.hh"
```

Funkcje

- void OtworzPlikIn (const char ∗nazwaPliku, std::fstream &plik)

  Otwiera plik do odczytu.
- void OtworzPlikOut (const char ∗nazwaPliku, std::fstream &plik)

  Otwiera plik do zapisu czyszcąc jego zawartość
- void LosujIntDoPliku (const unsigned int n, const unsigned int zakres)

  Zapisuje n losowych liczb(int) do pliku.

5.9.1 Opis szczegółowy

Plik zawiera definicje funkcji zwiazanych z obsluga plikow.

Definicja w pliku Pliki.cpp.

5.9.2 Dokumentacja funkcji

5.9.2.1 void LosujIntDoPliku (const unsigned int *n*, const unsigned int *zakres*)

Losuje n liczb z zakresu od 1 do podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parametry

in	*n*	- ilość liczb do zapisania
in	*zakres*	- górny zakres wartości liczb

Definicja w linii 27 pliku Pliki.cpp.

5.9.2.2 void OtworzPlikIn (const char * *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.9.2.3 void OtworzPlikOut (const char * *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.10 Dokumentacja pliku Pliki.hh

Funkcje obslugi plikow.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
```

Funkcje

- void OtworzPlikIn (const char *nazwaPliku, std::fstream &plik)

  Otwiera plik do odczytu.
- void OtworzPlikOut (const char *nazwaPliku, std::fstream &plik)

  Otwiera plik do zapisu czyszcąc jego zawartość
- void LosujIntDoPliku (const unsigned int n, const unsigned int zakres)

  Zapisuje n losowych liczb(int) do pliku.

5.10.1 Opis szczegółowy

Plik zawiera deklaracje funkcji zwiazanych z obsuga plikow

Definicja w pliku Pliki.hh.

5.10.2 Dokumentacja funkcji

5.10.2.1 void LosujIntDoPliku (const unsigned int *n*, const unsigned int *zakres*)

Losuje n liczb z zakresu od 1 do podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parametry

in	*n*	- ilość liczb do zapisania
in	*zakres*	- górny zakres wartości liczb

Definicja w linii 27 pliku Pliki.cpp.

5.10.2.2 void OtworzPlikIn (const char * *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.10.2.3 void OtworzPlikOut (const char * *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.11 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy Statystyka.

```
#include "../inc/Statystyka.hh"
```

5.11.1 Opis szczegółowy

Plik zawiera definicję metod klasy Statystyka.

Definicja w pliku Statystyka.cpp.

5.12 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy Statystyka.

```
#include <iostream>
#include "IObserwator.hh"
#include "Stoper.hh"
#include <fstream>
#include <cstdlib>
#include <string>
```

Komponenty

- class Statystyka

  Modeluje pojęcie statystyki.

5.12.1 Opis szczegółowy

Zawiera definicję klasy Statystyka

Definicja w pliku Statystyka.hh.

5.13 Dokumentacja pliku Stoper.cpp

```
#include "../inc/Stoper.hh"
```

5.14 Dokumentacja pliku Stoper.hh

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

Komponenty

- class Stoper

  Klasa Stoper.

5.15 Dokumentacja pliku TabAsoc.cpp

```
#include "../inc/TabAsoc.hh"
```

5.16 Dokumentacja pliku TabAsoc.hh

```
#include "TabHash.hh"
#include "Framework.hh"
```

Komponenty

- class TabAsoc

  Definicja klasy TabAsoc.

5.17 Dokumentacja pliku TabHash.cpp

```
#include "../inc/TabHash.hh"
```

5.18 Dokumentacja pliku TabHash.hh

```
#include <iostream>
#include "ListArr2x.hh"
```

Komponenty

- class TabHash

  Tablica Haszująca.
- struct TabHash::Para

  Para wartości klucz - wartość

Definicje

- #define ROZMIAR 1000033

  Definicja Tablicy Haszującej.

5.18.1 Dokumentacja definicji

5.18.1.1 #define ROZMIAR 1000033

Plik zawiera definicję Tablicy Haszującej

Definicja w linii 14 pliku TabHash.hh.

Skorowidz