# PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Cz, 16 kwi 2015 09:30:37

# 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>`

### Metody publiczne

- Benchmark (const unsigned int ileProb, unsigned int *const ileDanych, const unsigned int ilePowtorzen)
  *Konstruktor 2 argumentowy.*
- void Test (Framework *I, const char *const nazwaPlikuDane, std::string const nazwaPlikuStat) const
  *Testowanie algorytmu.*

### Atrybuty prywatne

- Statystyka * stat
  *Statystyki testu.*
- unsigned int IleProb
  *Ilość prób.*
- unsigned int * IleDanych

*Tablica liczności serii.*

- unsigned int IlePowtorzen

  *Ilość powtórzeń*

#### 4.1.1 Opis szczegółowy

**template<class typ>class Benchmark< typ >**

Modeluje pojęcie Benchmarku czyli objektu mierzącego czas wykonywania algoytmu

Definicja w linii 25 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 70 pliku Benchmark.hh.

#### 4.1.3 Dokumentacja funkcji składowych

##### 4.1.3.1 template<class typ> void Benchmark< typ >::Test ( Framework * *I,* const char *const *nazwaPlikuDane,* std::string const *nazwaPlikuStat* ) const [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 | *nazwaPlikuStat* | - nazwa pliku do którego zostaną zapisane statystyki param[in] nazwaPlikuDane - nazwa pliku z danymi neizbędnymi do przeprowadzenia testu |

Definicja w linii 88 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 48 pliku Benchmark.hh.

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

Ilość powtórzeń każdej serii

Definicja w linii 56 pliku Benchmark.hh.

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

Ilość powtórzeń każdej seriii

Definicja w linii 40 pliku Benchmark.hh.

#### 4.1.4.4 template<class typ> Statystyka* Benchmark< typ >::stat [private]

Pole przechowuje wyniki testów

Definicja w linii 32 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

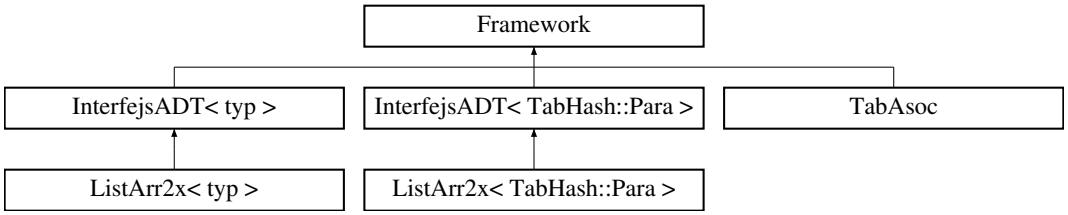
- Benchmark.hh

## 4.2 Dokumentacja klasy Framework

Modeluje interfejs programu.

`#include <Framework.hh>`

Diagram dziedziczenia dla Framework




### Metody publiczne

- virtual void Start (std::fstream &plik, const unsigned int k)=0

  *Wczytanie danych z pliku.*
- 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]

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

**Parametry**

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

Wykonanie części obliczeniowej programu

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

**Parametry**

| | | |
|---|---|---|
| in | *k* | - ilość elementów dla których mają zostać wykonane obliczenia. param[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::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.

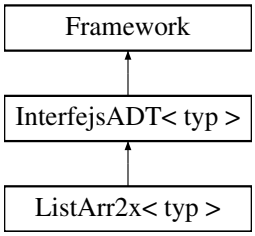
Dokumentacja dla tej klasy została wygenerowana z pliku:

- 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 typ pop (const unsigned int pole)=0

  *Pobiera element.*
- virtual unsigned int size () const =0

  *Liczność elemetów.*
- void Start (std::fstream &plik, const unsigned int k)=0

  *Wczytanie danych z pliku.*
- 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 typ 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]

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

**Parametry**

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

Wykonanie części obliczeniowej programu

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

**Parametry**

| | | |
|---|---|---|
| in | *k* | - ilość elementów dla których mają zostać wykonane obliczenia. param[in] plik<br>- plik z którego wczytujemy dane |

Implementuje Framework.

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

#### 4.3.2.5 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:

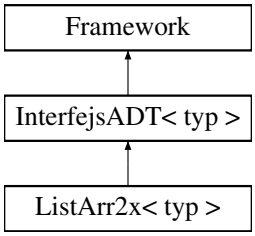
- InterfejsADT.hh

## 4.4 Dokumentacja 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 ListyArr1.*
- typ pop (const unsigned int pole)

  *Pobiera element z ListyArr1.*
- unsigned int size () const

  *Wielkość listy.*
- void Start (std::fstream &plik, const unsigned int k)

  *Metoda testująca czas.*
- void Zwolnij ()

  *Wczytuje dane z pliku.*
- typ operator[] (unsigned int i)
- typ & RefEnd ()
- void pokaz ()

### Atrybuty prywatne

- typ * tab

  *Wkaźnik na dynamiczną tablicę*
- unsigned int RozmiarT

  *Rozmiar tablicy.*
- unsigned int RozmiarL

  *Rozmiar Listy.*

### 4.4.1 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 18 pliku ListArr2x.hh.

#### 4.4.2 Dokumentacja konstruktora i destruktora

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

Kontruktor alokujący tablicę jednoelementową z której będzie tworzona lista

Definicja w linii 48 pliku ListArr2x.hh.

#### 4.4.3 Dokumentacja funkcji składowych

##### 4.4.3.1 template<class typ> typ ListArr2x< typ >::operator[] ( unsigned int *i* ) [inline]

Definicja w linii 198 pliku ListArr2x.hh.

##### 4.4.3.2 template<class typ> void ListArr2x< typ >::pokaz ( ) [inline]

Definicja w linii 209 pliku ListArr2x.hh.

##### 4.4.3.3 template<class typ> typ 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)

retval - zwraca wartosc pobranej danej lub '-1' w przyadku bledu

Implementuje InterfejsADT< typ >.

Definicja w linii 113 pliku ListArr2x.hh.

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

Dodaje nowy element do ListyArr1

**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 64 pliku ListArr2x.hh.

##### 4.4.3.5 template<class typ> typ& ListArr2x< typ >::RefEnd ( ) [inline]

Definicja w linii 202 pliku ListArr2x.hh.

##### 4.4.3.6 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 162 pliku ListArr2x.hh.

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

Metoda testująca czas wczytania n elementów na ListęArr1

**Parametry**

| | | |
|---|---|---|
| in | *k* | - ilość elementów do wczytania |

Implementuje InterfejsADT< typ >.

Definicja w linii 171 pliku ListArr2x.hh.

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

Wczytuje dane z pliku do ListArr1

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

Zwalnia pamięć

Zwalnia pamięć zaalokowaną przez ListArr2x

Implementuje InterfejsADT< typ >.

Definicja w linii 191 pliku ListArr2x.hh.

### 4.4.4 Dokumentacja atrybutów składowych

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

Aktualny rozmiar ListyArr2x

Definicja w linii 39 pliku ListArr2x.hh.

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

Aktualny rozmiar tablicy.

Definicja w linii 32 pliku ListArr2x.hh.

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

Wskaźnik na dynamiczną tablicę tworzącą ListęArr2x

Definicja w linii 25 pliku ListArr2x.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ListArr2x.hh

## 4.5 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*

### 4.5.1 Opis szczegółowy

Struktura modeluje nierozłączny element Tablicy Haszującej czyli parę klucz - wartość

Definicja w linii 31 pliku TabHash.hh.

### 4.5.2 Dokumentacja konstruktora i destruktora

#### 4.5.2.1 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.

#### 4.5.2.2 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.

#### 4.5.2.3 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.

### 4.5.3 Dokumentacja funkcji składowych

#### 4.5.3.1 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.

#### 4.5.4 Dokumentacja atrybutów składowych

##### 4.5.4.1 std::string TabHash::Para::Klucz

Klucz pod którym przechowywana jest wartość

Definicja w linii 39 pliku TabHash.hh.

##### 4.5.4.2 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

### 4.6 Dokumentacja klasy Statystyka

Modeluje pojęcie statystyki.

`#include <Statystyka.hh>`

#### Metody publiczne

- Statystyka (const unsigned int iloscProb, unsigned int ∗proby)
  *Konstruktor z dwoma pramametrami.*
- ∼Statystyka ()
  *Destruktor - zwaknia pamięć*
- double & operator[] (unsigned int i)
  *Indeksuje tablicę czasową*
- void ZapiszStaty (std::string nazwaPliku)
  *Zapisuje statysykę do pliku.*

#### Atrybuty prywatne

- unsigned int IleProb
  *Ilość prób.*
- unsigned int ∗ Proba
  *Tablica z rozmiarami prób.*
- double ∗ Czas
  *Średni czas wykonania danej próby.*

#### 4.6.1 Opis szczegółowy

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

Definicja w linii 22 pliku Statystyka.hh.

#### 4.6.2 Dokumentacja konstruktora i destruktora

##### 4.6.2.1 Statystyka::Statystyka ( const unsigned int *iloscProb,* unsigned int ∗ *proby* )

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. |

Definicja w linii 14 pliku Statystyka.cpp.

#### 4.6.2.2 Statystyka::∼Statystyka ( ) [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablicy przechowujące statystykę.

Definicja w linii 68 pliku Statystyka.hh.

### 4.6.3 Dokumentacja funkcji składowych

#### 4.6.3.1 double& Statystyka::operator[] ( unsigned int *i* ) [inline]

Zwraca referencję do i-tego indeksu tablicy czasowej.

**Parametry**

| | | |
|---|---|---|
| in | *i* | - indeks tablicy czasowej |

**Zwracane wartości**

| | |
|---|---|
| *Czas[i]* | referencja do wybranego indeksu |

Definicja w linii 80 pliku Statystyka.hh.

#### 4.6.3.2 void Statystyka::ZapiszStaty ( std::string *nazwaPliku* )

Zapisuje statystystykę do pliku o nazwie "statystyka.dat". Pierwsza linia pliku to wielkości prób druga to średnie czasy wykonania podane w ms;

Definicja w linii 22 pliku Statystyka.cpp.

### 4.6.4 Dokumentacja atrybutów składowych

#### 4.6.4.1 double∗ Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 46 pliku Statystyka.hh.

#### 4.6.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 30 pliku Statystyka.hh.

#### 4.6.4.3 unsigned int∗ Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 38 pliku Statystyka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

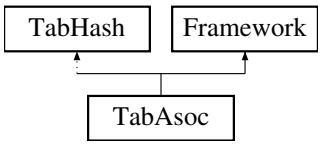
- Statystyka.hh
- Statystyka.cpp

## 4.7 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[].*
- const int Daj (const std::string klucz) const
- void Start (std::fstream &plik, const unsigned int k)

  *Wczytanie danych z pliku.*
- void Zwolnij ()

  *Zwalnia pamięć po teście.*

### Dodatkowe Dziedziczone Składowe

#### 4.7.1 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.

#### 4.7.2 Dokumentacja funkcji składowych

##### 4.7.2.1 const int TabAsoc::Daj ( const std::string *klucz* ) const

Definicja w linii 14 pliku TabAsoc.cpp.

##### 4.7.2.2 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.

#### 4.7.2.3 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 18 pliku TabAsoc.cpp.

#### 4.7.2.4 void TabAsoc::Start ( std::fstream & *plik,* const unsigned int *k* ) [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 |

Wykonanie części obliczeniowej programu

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

**Parametry**

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

Implementuje Framework.

Definicja w linii 26 pliku TabAsoc.cpp.

#### 4.7.2.5 void TabAsoc::Zwolnij ( ) [virtual]

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

Implementuje Framework.

Definicja w linii 22 pliku TabAsoc.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

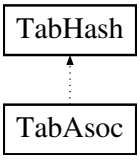
- TabAsoc.hh
- TabAsoc.cpp

## 4.8 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 ()
- ∼TabHash ()

#### 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.*

### 4.8.1 Opis szczegółowy

Klasa modeluje pojęcie Tablicy Haszującej

Definicja w linii 22 pliku TabHash.hh.

### 4.8.2 Dokumentacja konstruktora i destruktora

#### 4.8.2.1 TabHash::TabHash ( ) [protected]

Definicja w linii 67 pliku TabHash.cpp.

#### 4.8.2.2 TabHash::∼TabHash ( ) [protected]

Definicja w linii 72 pliku TabHash.cpp.

### 4.8.3 Dokumentacja funkcji składowych

#### 4.8.3.1 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 pzeszukania |
| in | *szukanyKlucz* | - klucz który ma zostać znaleziony |

**Zwracane wartości**

| | |
|---|---|
| - | zwraca wartość przechowywaną pod danym kluczem |

Definicja w linii 33 pliku TabHash.cpp.

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

Dodaje element (daną oraz jej klucz) do Tablicy Haszującej

**Parametry**

| | | |
|---|---|---|
| in | - | nowaDana - wartość którą dodajemy |
| in | - | nowyKlucz - klucz pod którym przechowujemy daną |

Definicja w linii 60 pliku TabHash.cpp.

#### 4.8.3.3 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.

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

Pobiera wartość przechowywaną pod zadanycm kluczem z Tablicy Haszującej

**Parametry**

| | | |
|---|---|---|
| in | *szukanyKlucz* | - klucz pod którym szukamy wartości |

**Zwracane wartości**

| | |
|---|---|
| - | zwraca wartość przechowywaną pod kluczem |

Definicja w linii 55 pliku TabHash.cpp.

### 4.8.4 Dokumentacja atrybutów składowych

#### 4.8.4.1 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 "Pliki.hh"
```

### 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 <fstream>
```

### 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 ListArr2x.hh

Definicja klasy ListArr1.

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

#### Komponenty

- class ListArr2x< typ >
  *Modeluje pojęcie Listy (array)*

### 5.4.1 Opis szczegółowy

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

Definicja w pliku ListArr2x.hh.

## 5.5 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 9
  *Ilość prób.*

#### Funkcje

- int main (int argc, char ∗argv[])

### 5.5.1 Opis szczegółowy

Program wkonuje serię 10 pomiarów czasu wykonania metody start dla różncyh wielkości problemu obliczeniowego, dla każdego zaimplemetowanego typu danych - LinkLista, ListaArr1, ListaArr2x. Procedura obliczeniowa polega na utworzeniu 'objektu' przechoującego n danych (stałych liczb). statystykę pomiarów zapisuje do pliku o nazwie "TypDaych.dat". gdzie "TypDanych" to odpowiednio Lista, ListaArr1 i ListaArr2x

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

Definicja w pliku main.cpp.

### 5.5.2 Dokumentacja definicji

#### 5.5.2.1 #define ILOSC_POWTORZEN 10

Ilośc powtórzeń danej próby

Definicja w linii 32 pliku main.cpp.

#### 5.5.2.2 #define ILOSC_PROB 9

Ilość prób = ilość rozmiarów prób

Definicja w linii 40 pliku main.cpp.

### 5.5.3 Dokumentacja funkcji

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

Definicja w linii 42 pliku main.cpp.

## 5.6 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.*

### 5.6.1 Opis szczegółowy

Plik zawiera definicje funkcji zwiazanych z obsluga plikow.

Definicja w pliku Pliki.cpp.

### 5.6.2 Dokumentacja funkcji

#### 5.6.2.1 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.

#### 5.6.2.2 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.

#### 5.6.2.3 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.

## 5.7 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.*

### 5.7.1 Opis szczegółowy

Plik zawiera deklaracje funkcji zwiazanych z obsuga plikow

Definicja w pliku Pliki.hh.

### 5.7.2 Dokumentacja funkcji

#### 5.7.2.1 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.

#### 5.7.2.2 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.

#### 5.7.2.3 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.

## 5.8 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy Statystyka.

```
#include "../inc/Statystyka.hh"
#include <fstream>
#include <cstdlib>
#include <string>
```

### 5.8.1 Opis szczegółowy

Plik zawiera definicję metod klasy Statystyka.

Definicja w pliku Statystyka.cpp.

## 5.9 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy Statystyka.

```
#include <iostream>
```

**Komponenty**

- class Statystyka

  *Modeluje pojęcie statystyki.*

### 5.9.1 Opis szczegółowy

Zawiera definicję klasy Statystyka

Definicja w pliku Statystyka.hh.

## 5.10 Dokumentacja pliku TabAsoc.cpp

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

## 5.11 Dokumentacja pliku TabAsoc.hh

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

### Komponenty

- class TabAsoc

  *Definicja klasy TabAsoc.*

## 5.12 Dokumentacja pliku TabHash.cpp

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

## 5.13 Dokumentacja pliku TabHash.hh

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

### Komponenty

- class TabHash

  *Tablica Haszująca.*
- struct TabHash::Para

  *Para wartości klucz - wartość*

### Definicje

- #define ROZMIAR 1

  *Definicja Tablicy Haszującej.*

#### 5.13.1 Dokumentacja definicji

##### 5.13.1.1 #define ROZMIAR 1

Plik zawiera definicję Tablicy Haszującej

Definicja w linii 14 pliku TabHash.hh.

## Skorowidz