

PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

So, 9 maj 2015 20:15:23

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	1
2.1 Lista klas	1
3 Indeks plików	2
3.1 Lista plików	2
4 Dokumentacja klas	3
4.1 Dokumentacja szablonu klasy Benchmark< typ >	3
4.1.1 Opis szczegółowy	4
4.1.2 Dokumentacja konstruktora i destruktora	4
4.1.3 Dokumentacja funkcji składowych	4
4.1.4 Dokumentacja atrybutów składowych	5
4.2 Dokumentacja klasy Framework	5
4.2.1 Opis szczegółowy	6
4.2.2 Dokumentacja funkcji składowych	6
4.3 Dokumentacja szablonu klasy InterfejsADT< typ >	7
4.3.1 Opis szczegółowy	7
4.3.2 Dokumentacja funkcji składowych	7
4.4 Dokumentacja klasy IObservator	9
4.4.1 Opis szczegółowy	10
4.4.2 Dokumentacja funkcji składowych	10
4.5 Dokumentacja klasy IObservowany	10
4.5.1 Opis szczegółowy	11
4.5.2 Dokumentacja funkcji składowych	11
4.6 Dokumentacja szablonu klasy ListArr2x< typ >	11
4.6.1 Opis szczegółowy	12
4.6.2 Dokumentacja konstruktora i destruktora	12
4.6.3 Dokumentacja funkcji składowych	12
4.6.4 Dokumentacja atrybutów składowych	14
4.7 Dokumentacja struktury TabHash::Para	14
4.7.1 Opis szczegółowy	14
4.7.2 Dokumentacja konstruktora i destruktora	15
4.7.3 Dokumentacja funkcji składowych	15
4.7.4 Dokumentacja atrybutów składowych	15
4.8 Dokumentacja klasy Statystyka	16
4.8.1 Opis szczegółowy	16

4.8.2	Dokumentacja konstruktora i destruktora	17
4.8.3	Dokumentacja funkcji składowych	18
4.8.4	Dokumentacja atrybutów składowych	18
4.9	Dokumentacja klasy Stoper	19
4.9.1	Opis szczegółowy	20
4.9.2	Dokumentacja konstruktora i destruktora	20
4.9.3	Dokumentacja funkcji składowych	20
4.9.4	Dokumentacja atrybutów składowych	20
4.10	Dokumentacja klasy TabAsoc	21
4.10.1	Opis szczegółowy	21
4.10.2	Dokumentacja funkcji składowych	22
4.11	Dokumentacja klasy TabHash	23
4.11.1	Opis szczegółowy	24
4.11.2	Dokumentacja konstruktora i destruktora	24
4.11.3	Dokumentacja funkcji składowych	24
4.11.4	Dokumentacja atrybutów składowych	26
5	Dokumentacja plików	26
5.1	Dokumentacja pliku Benchmark.hh	26
5.1.1	Opis szczegółowy	26
5.2	Dokumentacja pliku Framework.hh	26
5.2.1	Opis szczegółowy	27
5.3	Dokumentacja pliku InterfejsADT.hh	27
5.4	Dokumentacja pliku IObservator.hh	27
5.5	Dokumentacja pliku IObservowany.hh	27
5.5.1	Dokumentacja definicji	27
5.6	Dokumentacja pliku ListArr2x.hh	28
5.6.1	Opis szczegółowy	28
5.7	Dokumentacja pliku main.cpp	28
5.7.1	Opis szczegółowy	28
5.7.2	Dokumentacja definicji	29
5.7.3	Dokumentacja funkcji	29
5.8	Dokumentacja pliku Pliki.cpp	29
5.8.1	Opis szczegółowy	29
5.8.2	Dokumentacja funkcji	29
5.9	Dokumentacja pliku Pliki.hh	31
5.9.1	Opis szczegółowy	31
5.9.2	Dokumentacja funkcji	31
5.10	Dokumentacja pliku Statystyka.cpp	32
5.10.1	Opis szczegółowy	32

5.11 Dokumentacja pliku Statystyka.hh	32
5.11.1 Opis szczegółowy	32
5.12 Dokumentacja pliku Stoper.cpp	32
5.13 Dokumentacja pliku Stoper.hh	33
5.14 Dokumentacja pliku TabAsoc.cpp	33
5.15 Dokumentacja pliku TabAsoc.hh	33
5.16 Dokumentacja pliku TabHash.cpp	33
5.17 Dokumentacja pliku TabHash.hh	33
5.17.1 Dokumentacja definicji	34
Indeks	35

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

Framework	5
InterfejsADT< typ >	7
ListArr2x< typ >	11
InterfejsADT< TabHash::Para >	7
ListArr2x< TabHash::Para >	11
TabAsoc	21
IObserwator	9
Statystyka	16
IObserwowany	10
Benchmark< typ >	3
TabHash::Para	14
Stoper	19
TabHash	23
TabAsoc	21

2 Indeks klas

2.1 Lista klas

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

Benchmark< typ >	
Modeluje pojęcie Benchmarku	3
Framework	
Modeluje interfejs programu	5
InterfejsADT< typ >	7
IObserwator	
Klasa IObserwator	9
IObserwowany	
The IObserwowany class	10
ListArr2x< typ >	
Modeluje pojęcie Listy (array)	11
TabHash::Para	
Para wartości klucz - wartość	14
Statystyka	
Modeluje pojęcie statystyki	16
Stoper	
Klasa Stoper	19
TabAsoc	
Definicja klasy TabAsoc	21
TabHash	
Tablica Haszująca	23

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Benchmark.hh	
Definicja klasy Benchmark	26
Framework.hh	
Definicja klasy Framework	26
InterfejsADT.hh	27
IObserwator.hh	27
IObserwowany.hh	27
ListArr2x.hh	
Definicja klasy ListArr1	28
main.cpp	
Moduł główny programu	28
Pliki.cpp	
Definicje funkcji obsługi plików	29

Pliki.hh	
Funkcje obsługi plików	31
Statystyka.cpp	
Zawiera definicję metod klasy Statystyka	32
Statystyka.hh	
Zawiera definicję klasy Statystyka	32
Stoper.cpp	32
Stoper.hh	33
TabAsoc.cpp	33
TabAsoc.hh	33
TabHash.cpp	33
TabHash.hh	33

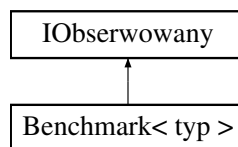
4 Dokumentacja klas

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](#) *f, 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 obiektu mierzącego czas wykonywania algorytmu

Definicja w linii 26 pliku Benchmark.hh.

4.1.2 Dokumentacja konstruktora i destruktor

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

Tworzy obiekt klasy [Benchmark](#) i inicjuje nową statystykę dla obiektu

Parametry

in	<i>ileProb</i>	- ilość prób, które zostaną wykonane
in	<i>ileDanych</i>	- wskaźnik na tablice z licznosciami kolejnych serii
in	<i>ilePowtorzen</i>	- 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 obiektu

Parametry

in	<i>nowyObserwator</i>	- wskaźnik na obiekt 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 określonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego przez użytkownika

Parametry

in	/	- obiekt klasy na której zostanie przeprowadzony test
in	<i>nazwaPlikuDane</i>	- nazwa pliku z danymi do wczytania

Definicja w linii 87 pliku Benchmark.hh.

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

Usuwa danego obserwatora z listy obserwatorów

Parametry

in	<i>obserwator</i>	- wskaźnik na obserwatora który ma zostać usunięty
----	-------------------	--

Implementuje [IObservowany](#).

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 licznosciami 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 serii

Definicja w linii 34 pliku Benchmark.hh.

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

Lista aktywnych obserwatorów danego obiektu

Definicja w linii 57 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

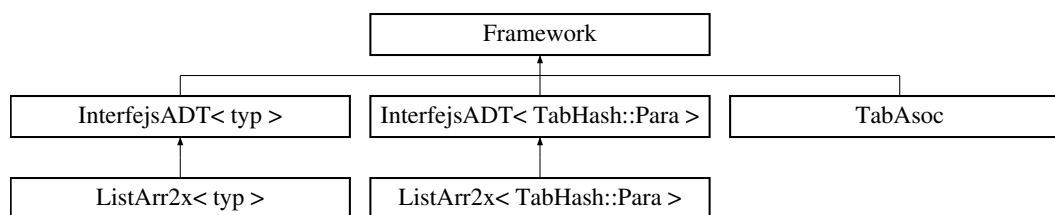
- [Benchmark.hh](#)

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 wykonywanych 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. 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::WczytajDane (const char * nazwaPliku, const unsigned int n) [pure virtual]

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

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku z danymi
in	<i>n</i>	- 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 obiekty wykorzystane do testów

Implementowany w [ListArr2x< typ >](#), [ListArr2x< TabHash::Para >](#), [InterfejsADT< typ >](#), [InterfejsADT< TabHash::Para >](#) i [TabAsoc](#).

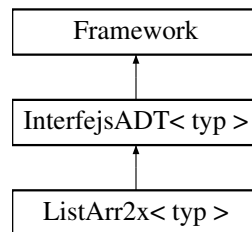
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [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 elementów.
- 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 typ InterfejsADT< typ >::pop (const unsigned int pole) [pure virtual]`

Pobiera element z typu danych

Parametry

<code>in</code>	<code>pole</code>	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola z ktore pobiera element
-----------------	-------------------	--

Zwracane wartości

<code>zwraca</code>	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	<i>dana</i>	- element który chcemy dorzucić do naszego typu
in	<i>pole</i>	- !!!DOSTĘPNE 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

<i>zwraca</i>	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	<i>k</i>	- ilość elementów dla których mają zostać wykonane obliczenia. param[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	<i>nazwaPliku</i>	- nazwa pliku z danymi
in	<i>n</i>	- 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:

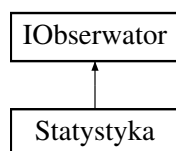
- [InterfejsADT.hh](#)

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 zawiera definicję klasy IObsereator.

The [IObserwator](#) class

Klasa modeluje interfejs obiektu 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 obiekcie obserwowanym.

Implementowany w [Statystyka](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

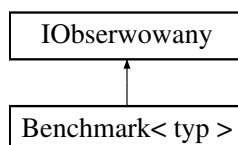
- [IObserwator.hh](#)

4.5 Dokumentacja klasy IObservowany

The [IObservowany](#) class.

```
#include <IObservowany.hh>
```

Diagram dziedziczenia dla IObservowany



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 obiektu obserwowanego.

Definicja w linii 17 pliku IObserwowany.hh.

4.5.2 Dokumentacja funkcji składowych

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

Dodaje nowego obserwatora do listy obserwatorów danego obiektu.

Parametry

in	<i>nowyObserwator</i>	- wskaźnik na dodawanego obserwatora
----	-----------------------	--------------------------------------

Implementowany w [Benchmark< typ >](#).

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

Powiadamia obserwatorów o wydarzeniu.

Implementowany w [Benchmark< typ >](#).

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

Usuwa danego obserwatora z listy obserwatorów danego obiektu.

Parametry

in	<i>obserwator</i>	- obserwator do usunięcia z listy
----	-------------------	-----------------------------------

Implementowany w [Benchmark< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

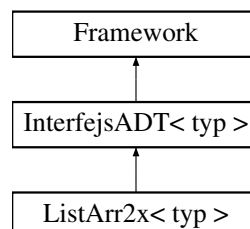
- [IObserwowany.hh](#)

4.6 Dokumentacja szablonu klasy ListArr2x< typ >

Modeluje pojęcie Listy (array)

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

Diagram dziedziczenia dla ListArr2x< typ >



Metody publiczne

- [ListArr2x](#) ()
Konstruktor bezargumentowy.
- 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 `WczytajDane` (const char *nazwaPliku, const unsigned int n)
Wczytuje dane z pliku.
- void `Zwolnij` ()
Zwalnia pamięć
- typ `operator[]` (unsigned int i)
- typ & `RefEnd` ()
- void `pokaz` ()

Atrybuty prywatne

- typ * `tab`
Wskaźnik na dynamiczną tablicę
- unsigned int `RozmiarT`
Rozmiar tablicy.
- unsigned int `RozmiarL`
Rozmiar Listy.

4.6.1 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.

4.6.2 Dokumentacja konstruktora i destruktora

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

Konstruktor alokujący tablicę jednoelementową z której będzie tworzona lista

Definicja w linii 48 pliku ListArr2x.hh.

4.6.3 Dokumentacja funkcji składowych

```
4.6.3.1 template<class typ> typ ListArr2x< typ >::operator[] ( unsigned int i ) [inline]
```

Definicja w linii 198 pliku ListArr2x.hh.

```
4.6.3.2 template<class typ> void ListArr2x< typ >::pokaz ( ) [inline]
```

Definicja w linii 209 pliku ListArr2x.hh.

4.6.3.3 `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 którego 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.

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

Dodaje nowy element do ListyArr1

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 64 pliku ListArr2x.hh.

4.6.3.5 `template<class typ> typ& ListArr2x< typ >::RefEnd () [inline]`

Definicja w linii 202 pliku ListArr2x.hh.

4.6.3.6 `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.

4.6.3.7 `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	<i>k</i>	- ilość elementów do wczytania
----	----------	--------------------------------

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 171 pliku ListArr2x.hh.

4.6.3.8 `template<class typ> void ListArr2x< typ >::WczytajDane (const char * nazwaPliku, const unsigned int n) [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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 184 pliku ListArr2x.hh.

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

Zwalnia pamięć zaalokowaną przez [ListArr2x](#)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 191 pliku ListArr2x.hh.

4.6.4 Dokumentacja atrybutów składowych

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

Aktualny rozmiar ListyArr2x

Definicja w linii 39 pliku ListArr2x.hh.

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

Aktualny rozmiar tablicy.

Definicja w linii 32 pliku ListArr2x.hh.

4.6.4.3 `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](#)

4.7 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 bezargumentowy.
- void [operator=](#) (const [Para](#) p)
Operator przypisania.

Atrybuty publiczne

- std::string [Klucz](#)
Klucz.
- int [Wartosc](#)
Wartość

4.7.1 Opis szczegółowy

Struktura modeluje nierozłączny element Tablicy Haszującej czyli parę klucz - wartość

Definicja w linii 31 pliku TabHash.hh.

4.7.2 Dokumentacja konstruktora i destruktor

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

Definicja metod Tablicy Haszującej.

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

Parametry

in	wart	- wartość, którą inicjujemy obiekt
	inf	key - klucz, którym inicjujemy obiekt

Plik zawiera definicję metod klasy [TabHash](#)

Definicja w linii 11 pliku TabHash.cpp.

4.7.2.2 TabHash::Para::Para (const int i)

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

Parametry

in	i	- wartosc którą zostanie zainicjowany obiekt
----	---	--

Definicja w linii 17 pliku TabHash.cpp.

4.7.2.3 TabHash::Para::Para ()

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

Definicja w linii 22 pliku TabHash.cpp.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 void TabHash::Para::operator= (const Para p)

Przeciążenie opratora przypisania - kopiuje i przypisuje wartości pól

Parametry

in	p	- obiekt który chcemy skopiować
----	---	---------------------------------

Definicja w linii 27 pliku TabHash.cpp.

4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 std::string TabHash::Para::Klucz

Klucz pod którym przechowywana jest wartość

Definicja w linii 39 pliku TabHash.hh.

4.7.4.2 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](#)

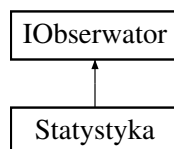
- [TabHash.cpp](#)

4.8 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 parametrami.
- [~Statystyka](#) ()
Destruktor - zwalnia pamięć
- void [ZapiszStaty](#) (std::string nazwaPliku) const
Zapisuje statystykę 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.

4.8.1 Opis szczegółowy

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

Definicja w linii 27 pliku Statystyka.hh.

4.8.2 Dokumentacja konstruktora i destruktora

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

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

Parametry

in	<i>iloscProb</i>	- liczba prob w ksperymentcie
in	<i>proby</i>	- tablica z licznosciami prób.

Definicja w linii 12 pliku Statystyka.cpp.

4.8.2.2 Statystyka::~Statystyka () [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablice przechowujące statystykę.

Definicja w linii 108 pliku Statystyka.hh.

4.8.3 Dokumentacja funkcji składowych**4.8.3.1 void Statystyka::Aktualizuj () [virtual]**

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

Implementuje [IObserwator](#).

Definicja w linii 44 pliku Statystyka.cpp.

4.8.3.2 void Statystyka::ZapiszStaty (std::string nazwaPliku) const

Zapisuje statystykę 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	<i>nazwaPliku</i>	- nazwa pliku do którego ma zostać zapisana statystyka
----	-------------------	--

Definicja w linii 25 pliku Statystyka.cpp.

4.8.4 Dokumentacja atrybutów składowych**4.8.4.1 double* Statystyka::Czas [private]**

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 51 pliku Statystyka.hh.

4.8.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 35 pliku Statystyka.hh.

4.8.4.3 unsigned int Statystyka::IloscPowtorzen [private]

Przechowuje ilość wykonywanych powtórzeń pojedynczego testu.

Definicja w linii 65 pliku Statystyka.hh.

4.8.4.4 unsigned int Statystyka::LicznikPowtorzen [private]

Zlicza ilość wykonanych powtórzeń w danej próbie.

Definicja w linii 72 pliku Statystyka.hh.

4.8.4.5 unsigned int Statystyka::LicznikProb [private]

Zlicza ilość prób wykonanych prób.

Definicja w linii 79 pliku Statystyka.hh.

4.8.4.6 Stoper* Statystyka::MojStoper [private]

[Stoper](#) wykorzystywany do pomiaru czasu.

Definicja w linii 86 pliku Statystyka.hh.

4.8.4.7 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 43 pliku Statystyka.hh.

4.8.4.8 double Statystyka::SumaCzasuProby [private]

Przechowuje sumę czasów pojedynczych powtórzeń z danej próby.

Definicja w linii 58 pliku Statystyka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Statystyka.hh](#)
- [Statystyka.cpp](#)

4.9 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 [CzasPoczątkowy](#)
Czas Początkowy.
- double [CzasKoncowy](#)
Czas Końcowy.
- bool [CzyLiczy](#)
Czy Liczy.

4.9.1 Opis szczegółowy

Plik zawiera definicję klasy [Stoper](#).

The [Stoper](#) class

Klasa modeluje stoper niezbędny do odliczania czasu.

Definicja w linii 18 pliku Stoper.hh.

4.9.2 Dokumentacja konstruktora i destruktora

4.9.2.1 Stoper::Stoper ()

Konstruktor bezargumentowy zeruje czasy i ustawia wartość pola CzyLiczy na false.

Definicja w linii 3 pliku Stoper.cpp.

4.9.3 Dokumentacja funkcji składowych

4.9.3.1 bool Stoper::CzyOdmierza () const

Informuje czy stoper aktualnie liczy czy nie.

Zwracane wartości

<i>true</i>	- gdy odlicza
<i>false</i>	- gdy nie odlicza

Definicja w linii 29 pliku Stoper.cpp.

4.9.3.2 double Stoper::DajPomiar () const

Wyłuskuje czas pomiaru w ms.

Zwracane wartości

<i>zwrca</i>	czas pomiaru wyrażon w ms
--------------	---------------------------

Definicja w linii 25 pliku Stoper.cpp.

4.9.3.3 void Stoper::Reset ()

Resetuje stoper.

Definicja w linii 19 pliku Stoper.cpp.

4.9.3.4 void Stoper::Start ()

Uruchamia odliczanie czasu.

Definicja w linii 9 pliku Stoper.cpp.

4.9.3.5 void Stoper::Stop ()

Zatrzymuje odliczanie czasu.

Definicja w linii 14 pliku Stoper.cpp.

4.9.4 Dokumentacja atrybutów składowych

4.9.4.1 double Stoper::CzasKoncowy [private]

Czas w którym odliczanie czasu zostało zatrzymane.

Definicja w linii 32 pliku Stoper.hh.

4.9.4.2 double Stoper::CzasPoczątkowy [private]

Czas w którym stoper zaczął odliczać.

Definicja w linii 25 pliku Stoper.hh.

4.9.4.3 bool Stoper::CzyLiczy [private]

Zmienna przechowuje wartość 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:

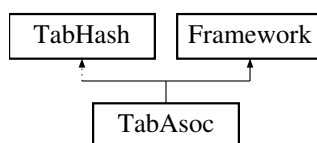
- [Stoper.hh](#)
- [Stoper.cpp](#)

4.10 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

4.10.1 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.

4.10.2 Dokumentacja funkcji składowych

4.10.2.1 `const int TabAsoc::operator() (const std::string klucz) const`

Definicje metod [TabAsoc](#).

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

Parametry

<code>in</code>	<code><i>klucz</i></code>	- klucz pod jakim chcemy znaleźć wartość
-----------------	---------------------------	--

Zwracane wartości

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

Plik zawiera definicje metod Tablicy Asocjacyjnej

Definicja w linii 10 pliku TabAsoc.cpp.

4.10.2.2 `int & TabAsoc::operator[] (const std::string klucz)`

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

Parametry

<code>in</code>	<code><i>klucz</i></code>	- klucz pod którym chcemy zapisać daną
-----------------	---------------------------	--

Zwracane wartości

	-	zwraca referencje do miejsca przechowywania danej
--	---	---

Definicja w linii 14 pliku TabAsoc.cpp.

4.10.2.3 `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 danych

Parametry

<code>in</code>	<code><i>plik</i></code>	- referencja do otwartego pliku z danymi
<code>in</code>	<code><i>k</i></code>	- ilość elementów na których zostanie przeprowadzony test

Implementuje [Framework](#).

Definicja w linii 46 pliku TabAsoc.cpp.

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

Wczytuje dane do Tablicy Haszującej z pliku, w którym linijka po linijce są podane kolejne wartości klucza, wartość.

[in] *nazwaPliku* - nazwa pliku z danymi

Parametry

<code>in</code>	<code><i>n</i></code>	- ilość danych do wczytania
-----------------	-----------------------	-----------------------------

Implementuje [Framework](#).

Definicja w linii 22 pliku TabAsoc.cpp.

4.10.2.5 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:

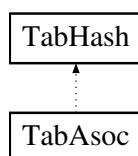
- [TabAsoc.hh](#)
- [TabAsoc.cpp](#)

4.11 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 element 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.

4.11.1 Opis szczegółowy

Klasa modeluje pojęcie Tablicy Haszującej

Definicja w linii 22 pliku TabHash.hh.

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 TabHash::TabHash () [protected]

Konstruktor bezargumentowy inicjuje tablicę pustymi listami

Definicja w linii 67 pliku TabHash.cpp.

4.11.2.2 TabHash::~~TabHash () [protected]

Destruktor - zwalnia pamięć po listach znajdujących się w tablicy.

Definicja w linii 72 pliku TabHash.cpp.

4.11.3 Dokumentacja funkcji składowych

4.11.3.1 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	<i>pozycja</i>	- pozycja Tablicy Haszującej na której znajduje się Lista do przeszukania
in	<i>szukanyKlucz</i>	- klucz który ma zostać znaleziony

Zwracane wartości

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

Definicja w linii 33 pliku TabHash.cpp.

4.11.3.2 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.

4.11.3.3 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	<i>klucz</i>	- klucz do haszowania
----	--------------	-----------------------

Zwracane wartości

-	zwraca numer indeksu Tablicy Haszującej
---	---

Definicja w linii 46 pliku TabHash.cpp.

4.11.3.4 `const int TabHash::Pobierz (const std::string szukanyKlucz) const` `[protected]`

Pobiera wartość przechowywaną pod zadanyim 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.

4.11.4 Dokumentacja atrybutów składowych**4.11.4.1 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 IObservator.hh

Komponenty

- class [IObservator](#)
Klasa [IObservator](#).

5.5 Dokumentacja pliku IObservowany.hh

```
#include "IObservator.hh"
```

Komponenty

- class [IObservowany](#)
The [IObservowany](#) 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 IObservowany.hh.

5.6 Dokumentacja pliku ListArr2x.hh

Definicja klasy ListArr1.

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

Komponenty

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

5.6.1 Opis szczegółowy

Plik zawiera definicję klasy ListArr2x ujętej w szablon typu wraz z jej składowymi metodami.

Definicja w pliku `ListArr2x.hh`.

5.7 Dokumentacja pliku main.cpp

Moduł główny programu.

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

Definicje

- `#define ILOSC_POWTORZEN 10`
Ilość powtórzeń danej próby.
- `#define ILOSC_PROB 8`
Ilość prób.

Funkcje

- `int main (int argc, char *argv[])`

5.7.1 Opis szczegółowy

Program wykonuje serię 10 pomiarów czasu wykonania metody start (badanie czasu zapisu i odczytu do/z Tablicy Asocjacyjnej dla różnych 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 sześciornakowymi ciągami stringów składających się wyłącznie z małych liter.

Wartości mogą być dowolnym intem

OBSŁUGA PROGRAMU: Aby wywołać program należy w linii poleceń wywołać jego nazwę np: `./a.out`

Definicja w pliku `main.cpp`.

5.7.2 Dokumentacja definicji

5.7.2.1 #define ILOSC_POWTORZEN 10

Ilość powtórzeń danej próby

Definicja w linii 39 pliku main.cpp.

5.7.2.2 #define ILOSC_PROB 8

Ilość prób = ilość rozmiarów prób

Definicja w linii 47 pliku main.cpp.

5.7.3 Dokumentacja funkcji

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

Definicja w linii 49 pliku main.cpp.

5.8 Dokumentacja pliku Pliki.cpp

Definicje funkcji obsługi plików.

```
#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 czyszcząc jego zawartość
- void [LosujIntDoPliku](#) (const unsigned int n, const unsigned int zakres)
Zapisuje n losowych liczb(int) do pliku.

5.8.1 Opis szczegółowy

Plik zawiera definicje funkcji związanych z obsługą plików.

Definicja w pliku [Pliki.cpp](#).

5.8.2 Dokumentacja funkcji

5.8.2.1 void LosujIntDoPliku (const unsigned int n, const unsigned int zakres)

Losuje n liczb z zakresu od 1 do podanego przez użytkownika następnie zapisuje wylosowane dane do pliku o nazwie "dane.dat"

Parametry

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

Definicja w linii 27 pliku Pliki.cpp.

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

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

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

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

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

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.9 Dokumentacja pliku Pliki.hh

Funkcje obsługi plików.

```
#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 czyszac 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 deklaracje funkcji związanych z obsługą plików

Definicja w pliku [Pliki.hh](#).

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 podanego przez użytkownika następnie zapisuje wylosowane dane do pliku o nazwie "dane.dat"

Parametry

in	<i>n</i>	- ilość liczb do zapisania
in	<i>zakres</i>	- 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	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyc
in	<i>plik</i>	- 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	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.10 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy [Statystyka](#).

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

5.10.1 Opis szczegółowy

Plik zawiera definicję metod klasy [Statystyka](#).

Definicja w pliku [Statystyka.cpp](#).

5.11 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy [Statystyka](#).

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

Komponenty

- class [Statystyka](#)
Modeluje pojęcie statystyki.

5.11.1 Opis szczegółowy

Zawiera definicję klasy [Statystyka](#)

Definicja w pliku [Statystyka.hh](#).

5.12 Dokumentacja pliku Stoper.cpp

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

5.13 Dokumentacja pliku Stoper.hh

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

Komponenty

- class [Stoper](#)
Klasa [Stoper](#).

5.14 Dokumentacja pliku TabAsoc.cpp

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

5.15 Dokumentacja pliku TabAsoc.hh

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

Komponenty

- class [TabAsoc](#)
Definicja klasy [TabAsoc](#).

5.16 Dokumentacja pliku TabHash.cpp

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

5.17 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.

5.17.1 Dokumentacja definicji

5.17.1.1 `#define ROZMIAR 1000033`

Plik zawiera definicję Tablicy Haszującej

Definicja w linii 14 pliku TabHash.hh.

Skorowidz

- ~Statystyka
 - Statystyka, 18
- ~TabHash
 - TabHash, 24
- _Tab
 - TabHash, 26
- Aktualizuj
 - IObserwator, 10
 - Statystyka, 18
- Benchmark
 - Benchmark, 4
 - DodajObserwatora, 4
 - IleDanych, 5
 - IlePowtorzen, 5
 - IleProb, 5
 - ListaObserwatorow, 5
 - PowiadomObserwatorow, 4
 - Test, 4
 - UsunObserwatora, 5
- Benchmark< typ >, 3
- Benchmark.hh, 26
- Czas
 - Statystyka, 18
- CzasKoncowy
 - Stoper, 20
- CzasPoczatkowy
 - Stoper, 21
- CzyLiczy
 - Stoper, 21
- CzyOdmierza
 - Stoper, 20
- DajPomiar
 - Stoper, 20
- DajZListy
 - TabHash, 24
- Dodaj
 - TabHash, 24
- DodajObserwatora
 - Benchmark, 4
 - IObserwowany, 11
- Framework, 5
 - Start, 6
 - WczytajDane, 6
 - Zwolnij, 6
- Framework.hh, 26
- H
 - TabHash, 24
- ILOSC_POWTORZEN
 - main.cpp, 29
- ILOSC_PROB
 - main.cpp, 29
- IOBSEWOWANY_HH
 - IObserwowany.hh, 27
- IObserwator, 9
 - Aktualizuj, 10
- IObserwator.hh, 27
- IObserwowany, 10
 - DodajObserwatora, 11
 - PowiadomObserwatorow, 11
 - UsunObserwatora, 11
- IObserwowany.hh, 27
- IOBSEWOWANY_HH, 27
- IleDanych
 - Benchmark, 5
- IlePowtorzen
 - Benchmark, 5
- IleProb
 - Benchmark, 5
 - Statystyka, 18
- IloscPowtorzen
 - Statystyka, 18
- InterfejsADT
 - pop, 7
 - push, 7
 - size, 9
 - Start, 9
 - WczytajDane, 9
 - Zwolnij, 9
- InterfejsADT< typ >, 7
- InterfejsADT.hh, 27
- Klucz
 - TabHash::Para, 15
- LicznikPowtorzen
 - Statystyka, 18
- LicznikProb
 - Statystyka, 18
- ListArr2x
 - ListArr2x, 12
 - ListArr2x, 12
 - pokaz, 12
 - pop, 12
 - push, 13
 - RefEnd, 13
 - RozmiarL, 14
 - RozmiarT, 14
 - size, 13
 - Start, 13
 - tab, 14
 - WczytajDane, 13
 - Zwolnij, 13
- ListArr2x< typ >, 11
- ListArr2x.hh, 28
- ListaObserwatorow
 - Benchmark, 5

LosujIntDoPliku
 Pliki.cpp, 29
 Pliki.hh, 31

 main
 main.cpp, 29
 main.cpp, 28
 ILOSC_POWTORZEN, 29
 ILOSC_PROB, 29
 main, 29
 MojStoper
 Statystyka, 19

 operator()
 TabAsoc, 22
 operator=
 TabHash::Para, 15
 OtworzPlikIn
 Pliki.cpp, 29
 Pliki.hh, 31
 OtworzPlikOut
 Pliki.cpp, 31
 Pliki.hh, 32

 Para
 TabHash::Para, 15
 Pliki.cpp, 29
 LosujIntDoPliku, 29
 OtworzPlikIn, 29
 OtworzPlikOut, 31
 Pliki.hh, 31
 LosujIntDoPliku, 31
 OtworzPlikIn, 31
 OtworzPlikOut, 32
 Pobierz
 TabHash, 24
 pokaz
 ListArr2x, 12
 pop
 InterfejsADT, 7
 ListArr2x, 12
 PowiadomObserwatorow
 Benchmark, 4
 IObservowany, 11
 Proba
 Statystyka, 19
 push
 InterfejsADT, 7
 ListArr2x, 13

 ROZMIAR
 TabHash.hh, 34
 RefEnd
 ListArr2x, 13
 Reset
 Stoper, 20
 RozmiarL
 ListArr2x, 14
 RozmiarT
 ListArr2x, 14

 size
 InterfejsADT, 9
 ListArr2x, 13
 Start
 Framework, 6
 InterfejsADT, 9
 ListArr2x, 13
 Stoper, 20
 TabAsoc, 22
 Statystyka, 16
 ~Statystyka, 18
 Aktualizuj, 18
 Czas, 18
 IleProb, 18
 IloscPowtorzen, 18
 LicznikPowtorzen, 18
 LicznikProb, 18
 MojStoper, 19
 Proba, 19
 Statystyka, 17
 SumaCzasuProby, 19
 ZapiszStaty, 18
 Statystyka.cpp, 32
 Statystyka.hh, 32
 Stop
 Stoper, 20
 Stoper, 19
 CzasKoncowy, 20
 CzasPoczkowy, 21
 CzyLiczy, 21
 CzyOdmierza, 20
 DajPomiar, 20
 Reset, 20
 Start, 20
 Stop, 20
 Stoper, 20
 Stoper.cpp, 32
 Stoper.hh, 33
 SumaCzasuProby
 Statystyka, 19

 tab
 ListArr2x, 14
 TabAsoc, 21
 operator(), 22
 Start, 22
 WczytajDane, 22
 Zwolnij, 22
 TabAsoc.cpp, 33
 TabAsoc.hh, 33
 TabHash, 23
 ~TabHash, 24
 _Tab, 26
 DajZListy, 24
 Dodaj, 24
 H, 24
 Pobierz, 24

- TabHash, [24](#)
- TabHash, [24](#)
- TabHash.cpp, [33](#)
- TabHash.hh, [33](#)
- ROZMIAR, [34](#)
- TabHash::Para, [14](#)
 - Klucz, [15](#)
 - operator=, [15](#)
 - Para, [15](#)
 - Wartosc, [15](#)
- Test
 - Benchmark, [4](#)
- UsunObserwatora
 - Benchmark, [5](#)
 - IObservowany, [11](#)
- Wartosc
 - TabHash::Para, [15](#)
- WczytajDane
 - Framework, [6](#)
 - InterfejsADT, [9](#)
 - ListArr2x, [13](#)
 - TabAsoc, [22](#)
- ZapiszStaty
 - Statystyka, [18](#)
- Zwolnij
 - Framework, [6](#)
 - InterfejsADT, [9](#)
 - ListArr2x, [13](#)
 - TabAsoc, [22](#)