

PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

So, 18 kwi 2015 21:49:12

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	2
2.1 Lista klas	2
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	4
4.2 Dokumentacja klasy Framework	5
4.2.1 Opis szczegółowy	5
4.2.2 Dokumentacja funkcji składowych	5
4.3 Dokumentacja szablonu klasy InterfejsADT< typ >	7
4.3.1 Opis szczegółowy	7
4.3.2 Dokumentacja funkcji składowych	8
4.4 Dokumentacja szablonu klasy ListArr2x< typ >	9
4.4.1 Opis szczegółowy	10
4.4.2 Dokumentacja konstruktora i destruktora	10
4.4.3 Dokumentacja funkcji składowych	10
4.4.4 Dokumentacja atrybutów składowych	11
4.5 Dokumentacja struktury TabHash::Para	12
4.5.1 Opis szczegółowy	12
4.5.2 Dokumentacja konstruktora i destruktora	12
4.5.3 Dokumentacja funkcji składowych	13
4.5.4 Dokumentacja atrybutów składowych	13
4.6 Dokumentacja klasy Statystyka	13
4.6.1 Opis szczegółowy	14
4.6.2 Dokumentacja konstruktora i destruktora	14
4.6.3 Dokumentacja funkcji składowych	14
4.6.4 Dokumentacja atrybutów składowych	14
4.7 Dokumentacja klasy TabAsoc	15
4.7.1 Opis szczegółowy	15
4.7.2 Dokumentacja funkcji składowych	16

4.8	Dokumentacja klasy TabHash	17
4.8.1	Opis szczegółowy	18
4.8.2	Dokumentacja konstruktora i destruktora	18
4.8.3	Dokumentacja funkcji składowych	18
4.8.4	Dokumentacja atrybutów składowych	20
5	Dokumentacja plików	20
5.1	Dokumentacja pliku Benchmark.hh	20
5.1.1	Opis szczegółowy	20
5.2	Dokumentacja pliku Framework.hh	20
5.2.1	Opis szczegółowy	21
5.3	Dokumentacja pliku InterfejsADT.hh	21
5.4	Dokumentacja pliku ListArr2x.hh	21
5.4.1	Opis szczegółowy	21
5.5	Dokumentacja pliku main.cpp	21
5.5.1	Opis szczegółowy	22
5.5.2	Dokumentacja definicji	22
5.5.3	Dokumentacja funkcji	22
5.6	Dokumentacja pliku Pliki.cpp	22
5.6.1	Opis szczegółowy	23
5.6.2	Dokumentacja funkcji	23
5.7	Dokumentacja pliku Pliki.hh	23
5.7.1	Opis szczegółowy	24
5.7.2	Dokumentacja funkcji	24
5.8	Dokumentacja pliku Statystyka.cpp	24
5.8.1	Opis szczegółowy	24
5.9	Dokumentacja pliku Statystyka.hh	24
5.9.1	Opis szczegółowy	25
5.10	Dokumentacja pliku TabAsoc.cpp	25
5.11	Dokumentacja pliku TabAsoc.hh	25
5.12	Dokumentacja pliku TabHash.cpp	25
5.13	Dokumentacja pliku TabHash.hh	25
5.13.1	Dokumentacja definicji	26
Indeks		27

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

Benchmark< typ >	3
Framework	5
InterfejsADT< typ >	7
ListArr2x< typ >	9
InterfejsADT< TabHash::Para >	7
ListArr2x< TabHash::Para >	9
TabAsoc	15
TabHash::Para	12
Statystyka	13
TabHash	17
TabAsoc	15

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
ListArr2x< typ >	
Modeluje pojęcie Listy (array)	9
TabHash::Para	
Para wartości klucz - wartość	12
Statystyka	
Modeluje pojęcie statystyki	13
TabAsoc	
Definicja klasy TabAsoc	15
TabHash	
Tablica Haszująca	17

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Benchmark.hh	
Definicja klasy Benchmark	20
Framework.hh	
Definicja klasy Framework	20
InterfejsADT.hh	21
ListArr2x.hh	
Definicja klasy ListArr1	21
main.cpp	
Moduł główny programu	21
Pliki.cpp	
Definicje funkcji obsługi plików	22
Pliki.hh	
Funkcje obsługi plików	23
Statystyka.cpp	
Zawiera definicję metod klasy Statystyka	24
Statystyka.hh	
Zawiera definicję klasy Statystyka	24
TabAsoc.cpp	25
TabAsoc.hh	25
TabHash.cpp	25
TabHash.hh	25

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](#) *, 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 obiektu 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 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 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 określonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego przez użytkownika

Parametry

in	<i>I</i>	- obiekt klasy na której zostanie przeprowadzony test
in	<i>nazwaPlikuStat</i>	- nazwa pliku do którego zostaną zapisane statystyki param[in] <i>nazwaPlikuDane</i> - nazwa pliku z danymi niezbędnymi 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 licznosciami elementów dla kolejnych 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 serii

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:

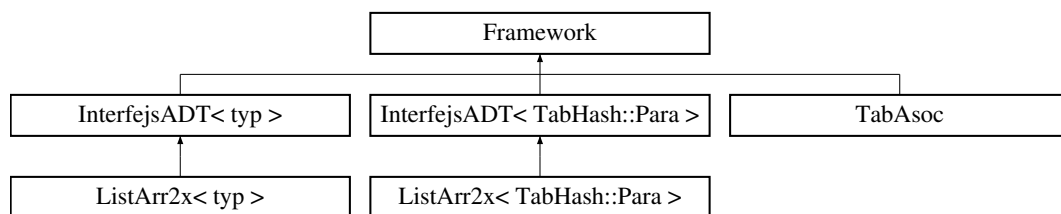
- [Benchmark.hh](#)

4.2 Dokumentacja klasy Framework

Modeluje interfejs programu.

```
#include <Framework.hh>
```

Diagram dziedziczenia dla Framework



Metody publiczne

- virtual void [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](#).

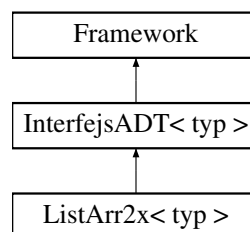
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [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

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

Zwracane wartości

<i>zwraca</i>	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

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

<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

<i>in</i>	<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

<i>in</i>	<i>nazwaPliku</i>	- nazwa pliku z danymi
<i>in</i>	<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:

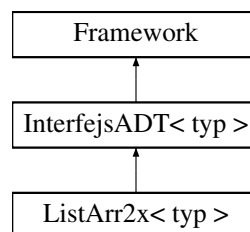
- [InterfejsADT.hh](#)

4.4 Dokumentacja 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.

4.4.1 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]`

Konstruktor 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 którego chcemy pobrać element (indeksowane od 0)

retval - zwraca wartość pobranej danej lub '-1' w przypadku błędu

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	<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.

4.4.3.5 `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

<i>in</i>	<i>k</i>	- ilość elementów do wczytania
-----------	----------	--------------------------------

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 171 pliku ListArr2x.hh.

4.4.3.8 `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.

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

Zwalnia pamięć zaalokowaną przez [ListArr2x](#)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 191 pliku ListArr2x.hh.

4.4.4 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 bezargumentowy.
- 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 destruktor

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 obiekt inicjując go podanymi wartościami

Parametry

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

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 (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.

4.5.2.3 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.

4.5.3 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	- obiekt 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 parametrami.
- [~Statystyka](#) ()
Destruktor - zwalnia pamięć
- double & [operator\[\]](#) (unsigned int i)
Indeksuje tablicę czasową
- void [ZapiszStaty](#) (std::string nazwaPliku)
Zapisuje statystykę 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 różnych wielkości prób.

Definicja w linii 22 pliku Statystyka.hh.

4.6.2 Dokumentacja konstruktora i destruktor

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

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

Parametry

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

Definicja w linii 14 pliku Statystyka.cpp.

4.6.2.2 Statystyka::~~Statystyka () [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablice 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

<code>in</code>	<i>i</i>	- indeks tablicy czasowej
-----------------	----------	---------------------------

Zwracane wartości

<i>Czas[i]</i>	referencja do wybranego indeksu
----------------	---------------------------------

Definicja w linii 80 pliku Statystyka.hh.

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

Zapisuje statystykę 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:

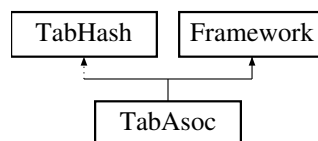
- [Statystyka.hh](#)
- [Statystyka.cpp](#)

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[].
- `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

4.7.1 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::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>klucz</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.

4.7.2.2 `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>klucz</code>	- klucz pod którym chcemy zapisać daną
-----------------	--------------------	----------------------------------------

Zwracane wartości

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

Definicja w linii 14 pliku TabAsoc.cpp.

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

Implementuje [Framework](#).

Definicja w linii 46 pliku TabAsoc.cpp.

4.7.2.4 `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, wartości.

[in] nazwaPliku - nazwa pliku z danymi

Parametry

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

Implementuje [Framework](#).

Definicja w linii 22 pliku TabAsoc.cpp.

4.7.2.5 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:

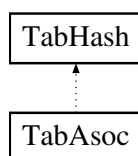
- [TabAsoc.hh](#)
- [TabAsoc.cpp](#)

4.8 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 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.

4.8.1 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]

Konstruktor bezargumentowy inicjuje tablicę pustymi listami

Definicja w linii 67 pliku TabHash.cpp.

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

Destruktor - zwalnia pamięć po listach znajdujących się w tablicy.

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	<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.

4.8.3.2 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	<i>klucz</i>	- 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 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.

4.8.4 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 "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 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 ListArr2x ujętej w szablon typu wraz z jej składowymi metodami.

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`
Ilość powtórzeń danej próby.
- `#define ILOSC_PROB 7`
Ilość prób.

Funkcje

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

5.5.1 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 sześciocyfrowymi ciągami stringów składających się wyłącznie z małych liter.

Wartości mogą być dowolnym intem

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

Definicja w pliku `main.cpp`.

5.5.2 Dokumentacja definicji

5.5.2.1 `#define ILOSC_POWTORZEN 10`

Ilość powtórzeń danej próby

Definicja w linii 39 pliku `main.cpp`.

5.5.2.2 `#define ILOSC_PROB 7`

Ilość prób = ilość rozmiarów prób

Definicja w linii 47 pliku `main.cpp`.

5.5.3 Dokumentacja funkcji

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

Definicja w linii 49 pliku `main.cpp`.

5.6 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.

5.6.1 Opis szczegółowy

Plik zawiera definicje funkcji związanych z obsługą plików.

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 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.

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

Otwiera plik i sprawdza czy otwarcie się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyć
in	<i>plik</i>	- strumień powiązany 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 się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyć
in	<i>plik</i>	- strumień powiązany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.7 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 czyszcząc 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 związanych z obsługą plików

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 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.

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

Otwiera plik i sprawdza czy otwarcie się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyć
in	<i>plik</i>	- strumień powiązany 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 się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyć
in	<i>plik</i>	- strumień powiązany 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 1000033`
Definicja Tablicy Haszującej.

5.13.1 Dokumentacja definicji

5.13.1.1 `#define ROZMIAR 1000033`

Plik zawiera definicję Tablicy Haszującej

Definicja w linii 14 pliku TabHash.hh.

Skorowidz

- ~Statystyka
 - Statystyka, [14](#)
- ~TabHash
 - TabHash, [18](#)
- _Tab
 - TabHash, [20](#)
- Benchmark
 - Benchmark, [4](#)
 - IleDanych, [4](#)
 - IlePowtorzen, [4](#)
 - IleProb, [4](#)
 - stat, [5](#)
 - Test, [4](#)
- Benchmark< typ >, [3](#)
- Benchmark.hh, [20](#)
- Czas
 - Statystyka, [14](#)
- DajZListy
 - TabHash, [18](#)
- Dodaj
 - TabHash, [18](#)
- Framework, [5](#)
 - Start, [5](#)
 - WczytajDane, [5](#)
 - Zwolnij, [7](#)
- Framework.hh, [20](#)
- H
 - TabHash, [18](#)
- ILOSC_POWTORZEN
 - main.cpp, [22](#)
- ILOSC_PROB
 - main.cpp, [22](#)
- IleDanych
 - Benchmark, [4](#)
- IlePowtorzen
 - Benchmark, [4](#)
- IleProb
 - Benchmark, [4](#)
 - Statystyka, [15](#)
- InterfejsADT
 - pop, [8](#)
 - push, [8](#)
 - size, [8](#)
 - Start, [8](#)
 - WczytajDane, [8](#)
 - Zwolnij, [8](#)
- InterfejsADT< typ >, [7](#)
- InterfejsADT.hh, [21](#)
- Klucz
 - TabHash::Para, [13](#)

- ListArr2x
 - ListArr2x, [10](#)
 - ListArr2x, [10](#)
 - pokaz, [10](#)
 - pop, [10](#)
 - push, [10](#)
 - RefEnd, [10](#)
 - RozmiarL, [11](#)
 - RozmiarT, [11](#)
 - size, [10](#)
 - Start, [11](#)
 - tab, [11](#)
 - WczytajDane, [11](#)
 - Zwolnij, [11](#)
- ListArr2x< typ >, [9](#)
- ListArr2x.hh, [21](#)
- LosujIntDoPliku
 - Pliki.cpp, [23](#)
 - Pliki.hh, [24](#)
- main
 - main.cpp, [22](#)
- main.cpp, [21](#)
 - ILOSC_POWTORZEN, [22](#)
 - ILOSC_PROB, [22](#)
 - main, [22](#)
- operator()
 - TabAsoc, [16](#)
- operator=
 - TabHash::Para, [13](#)
- OtworzPlikIn
 - Pliki.cpp, [23](#)
 - Pliki.hh, [24](#)
- OtworzPlikOut
 - Pliki.cpp, [23](#)
 - Pliki.hh, [24](#)
- Para
 - TabHash::Para, [12](#)
- Pliki.cpp, [22](#)
 - LosujIntDoPliku, [23](#)
 - OtworzPlikIn, [23](#)
 - OtworzPlikOut, [23](#)
- Pliki.hh, [23](#)
 - LosujIntDoPliku, [24](#)
 - OtworzPlikIn, [24](#)
 - OtworzPlikOut, [24](#)
- Pobierz
 - TabHash, [18](#)
- pokaz
 - ListArr2x, [10](#)
- pop
 - InterfejsADT, [8](#)
 - ListArr2x, [10](#)
- Proba

- Statystyka, 15
- push
 - InterfejsADT, 8
 - ListArr2x, 10
- ROZMIAR
 - TabHash.hh, 26
- RefEnd
 - ListArr2x, 10
- RozmiarL
 - ListArr2x, 11
- RozmiarT
 - ListArr2x, 11
- size
 - InterfejsADT, 8
 - ListArr2x, 10
- Start
 - Framework, 5
 - InterfejsADT, 8
 - ListArr2x, 11
 - TabAsoc, 16
- stat
 - Benchmark, 5
- Statystyka, 13
 - ~Statystyka, 14
 - Czas, 14
 - IleProb, 15
 - Proba, 15
 - Statystyka, 14
 - ZapiszStaty, 14
- Statystyka.cpp, 24
- Statystyka.hh, 24
- tab
 - ListArr2x, 11
- TabAsoc, 15
 - operator(), 16
 - Start, 16
 - WczytajDane, 16
 - Zwolnij, 16
- TabAsoc.cpp, 25
- TabAsoc.hh, 25
- TabHash, 17
 - ~TabHash, 18
 - _Tab, 20
 - DajZListy, 18
 - Dodaj, 18
 - H, 18
 - Pobierz, 18
 - TabHash, 18
 - TabHash, 18
- TabHash.cpp, 25
- TabHash.hh, 25
 - ROZMIAR, 26
- TabHash::Para, 12
 - Klucz, 13
 - operator=, 13
 - Para, 12
 - Wartosc, 13
- Test
 - Benchmark, 4
- Wartosc
 - TabHash::Para, 13
- WczytajDane
 - Framework, 5
 - InterfejsADT, 8
 - ListArr2x, 11
 - TabAsoc, 16
- ZapiszStaty
 - Statystyka, 14
- Zwolnij
 - Framework, 7
 - InterfejsADT, 8
 - ListArr2x, 11
 - TabAsoc, 16