

PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Cz, 14 maj 2015 08:14:37

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	3
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	5
4.1.4 Dokumentacja atrybutów składowych	5
4.2 Dokumentacja klasy Framework	6
4.2.1 Opis szczegółowy	6
4.2.2 Dokumentacja funkcji składowych	6
4.3 Dokumentacja szablonu klasy HeapSort< typ >	7
4.3.1 Opis szczegółowy	8
4.3.2 Dokumentacja funkcji składowych	8
4.4 Dokumentacja szablonu klasy HybridSort< typ >	8
4.4.1 Opis szczegółowy	9
4.4.2 Dokumentacja funkcji składowych	9
4.5 Dokumentacja szablonu klasy InsertSort< typ >	9
4.5.1 Opis szczegółowy	10
4.5.2 Dokumentacja funkcji składowych	10
4.6 Dokumentacja szablonu klasy InterfejsADT< typ >	10
4.6.1 Opis szczegółowy	11
4.6.2 Dokumentacja funkcji składowych	11
4.7 Dokumentacja klasy IObservator	13
4.7.1 Opis szczegółowy	13
4.7.2 Dokumentacja funkcji składowych	14
4.8 Dokumentacja klasy IObservowany	14
4.8.1 Opis szczegółowy	14
4.8.2 Dokumentacja funkcji składowych	14
4.9 Dokumentacja szablonu klasy ISort< typ >	15
4.9.1 Opis szczegółowy	15
4.9.2 Dokumentacja funkcji składowych	15

4.10	Dokumentacja szablonu klasy <code>Iterable< typ ></code>	16
4.10.1	Opis szczegółowy	16
4.10.2	Dokumentacja funkcji składowych	16
4.11	Dokumentacja szablonu klasy <code>ListArr2x< typ ></code>	17
4.11.1	Opis szczegółowy	18
4.11.2	Dokumentacja konstruktora i destruktor	18
4.11.3	Dokumentacja funkcji składowych	18
4.11.4	Dokumentacja atrybutów składowych	21
4.12	Dokumentacja szablonu klasy <code>QuickSortOpt< typ ></code>	22
4.12.1	Opis szczegółowy	22
4.12.2	Dokumentacja funkcji składowych	22
4.12.3	Dokumentacja przyjaciół i funkcji związanych	23
4.13	Dokumentacja klasy <code>Statystyka</code>	23
4.13.1	Opis szczegółowy	24
4.13.2	Dokumentacja konstruktora i destruktor	24
4.13.3	Dokumentacja funkcji składowych	25
4.13.4	Dokumentacja atrybutów składowych	25
4.14	Dokumentacja klasy <code>Stoper</code>	26
4.14.1	Opis szczegółowy	27
4.14.2	Dokumentacja konstruktora i destruktor	27
4.14.3	Dokumentacja funkcji składowych	27
4.14.4	Dokumentacja atrybutów składowych	27
5	Dokumentacja plików	28
5.1	Dokumentacja pliku <code>Benchmark.hh</code>	28
5.1.1	Opis szczegółowy	28
5.2	Dokumentacja pliku <code>Framework.hh</code>	28
5.2.1	Opis szczegółowy	29
5.3	Dokumentacja pliku <code>HeapSort.hh</code>	29
5.4	Dokumentacja pliku <code>HybridSort.hh</code>	29
5.5	Dokumentacja pliku <code>InsertSort.hh</code>	29
5.6	Dokumentacja pliku <code>InterfejsADT.hh</code>	29
5.7	Dokumentacja pliku <code>IObserwator.hh</code>	30
5.8	Dokumentacja pliku <code>IObserwowany.hh</code>	30
5.8.1	Dokumentacja definicji	30
5.9	Dokumentacja pliku <code>ISort.hh</code>	30
5.10	Dokumentacja pliku <code>Iterable.hh</code>	30
5.10.1	Dokumentacja definicji	31
5.11	Dokumentacja pliku <code>ListArr2x.hh</code>	31
5.11.1	Opis szczegółowy	31

5.12 Dokumentacja pliku main.cpp	31
5.12.1 Opis szczegółowy	32
5.12.2 Dokumentacja funkcji	32
5.12.3 Dokumentacja zmiennych	32
5.13 Dokumentacja pliku Pliki.cpp	32
5.13.1 Opis szczegółowy	32
5.13.2 Dokumentacja funkcji	32
5.14 Dokumentacja pliku Pliki.hh	34
5.14.1 Opis szczegółowy	34
5.14.2 Dokumentacja funkcji	34
5.15 Dokumentacja pliku QuickSortOpt.hh	35
5.16 Dokumentacja pliku Statystyka.cpp	35
5.16.1 Opis szczegółowy	35
5.17 Dokumentacja pliku Statystyka.hh	35
5.17.1 Opis szczegółowy	36
5.18 Dokumentacja pliku Stoper.cpp	36
5.19 Dokumentacja pliku Stoper.hh	36
Indeks	37

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

Framework	6
InterfejsADT< typ >	10
ListArr2x< typ >	17
IObserwator	13
Statystyka	23
IObserwowany	14
Benchmark< typ >	3
ISort< typ >	15
HeapSort< typ >	7
HybridSort< typ >	8
InsertSort< typ >	9
QuickSortOpt< typ >	22

Iterable< typ >	16
ListArr2x< typ >	17
Stoper	26

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	6
HeapSort< typ > Heap Sort	7
HybridSort< typ > Definicja klasy HybridSort	8
InsertSort< typ > Insert Sort	9
InterfejsADT< typ >	10
IObserwator Klasa IObserwator	13
IObserwowany Interfejs obserwowanego	14
ISort< typ > Interfejs ISort	15
Iterable< typ > Interfejs Iterable	16
ListArr2x< typ > Modeluje pojęcie Listy (array)	17
QuickSortOpt< typ > Definicja klasy QuickSortOpt	22
Statystyka Modeluje pojęcie statystyki	23
Stoper Klasa Stoper	26

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Benchmark.hh	
Definicja klasy Benchmark	28
Framework.hh	
Definicja klasy Framework	28
HeapSort.hh	29
HybridSort.hh	29
InsertSort.hh	29
InterfejsADT.hh	29
IObserwator.hh	30
IObserwowany.hh	30
ISort.hh	30
Iterable.hh	30
ListArr2x.hh	
Definicja klasy ListArr2x	31
main.cpp	
Moduł główny programu	31
Pliki.cpp	
Definicje funkcji obsługi plików	32
Pliki.hh	
Funkcje obsługi plików	34
QuickSortOpt.hh	35
Statystyka.cpp	
Zawiera definicję metod klasy Statystyka	35
Statystyka.hh	
Zawiera definicję klasy Statystyka	35
Stoper.cpp	36
Stoper.hh	36

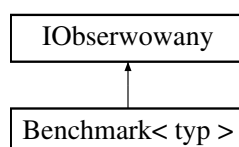
4 Dokumentacja klas

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** (InterfejsADT< typ > &I, std::string const nazwaPliku, ISort< typ > &sort, Iterable< typ > &IT)
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 algoytmu

Definicja w linii 27 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 72 pliku Benchmark.hh.

4.1.3 Dokumentacja funkcji składowych

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

Dodaje obserwatora do listy obserwatorów danego obiektu

Parametry

in	<i>nowyObserwator</i>	- wskaźnik na obiekt będący obserwatorem
----	-----------------------	--

Implementuje [IObservowany](#).

Definicja w linii 111 pliku Benchmark.hh.

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

Wywołuje u wszystkich aktywnych obserwatorów metodę Aktualizuj.

Implementuje [IObservowany](#).

Definicja w linii 131 pliku Benchmark.hh.

4.1.3.3 `template<class typ> void Benchmark< typ >::Test (InterfejsADT< typ > & I, std::string const nazwaPliku,`
`ISort< typ > & sort, Iterable< typ > & IT)` `[inline]`

Metoda testuje algorytm w określonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego przez użytkownika

Parametry

in	<i>I</i>	- obiekt klasy implementującej InterfejsADT na której zostanie przeprowadzony test
in	<i>nazwaPliku</i>	- nazwa pliku z danymi do wczytania
in	<i>sort</i>	- referencja do obiektu sortującego
in	<i>IT</i>	- referencja do obiektu implementującego Iterable

Definicja w linii 90 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 122 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 43 pliku Benchmark.hh.

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

Ilość powtórzeń każdej serii

Definicja w linii 51 pliku Benchmark.hh.

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

Ilość powtórzeń każdej serii

Definicja w linii 35 pliku Benchmark.hh.

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

Lista aktywnych obserwatorów danego obiektu

Definicja w linii 58 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

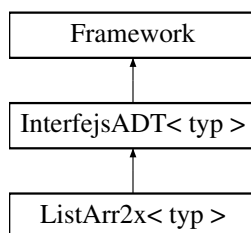
- [Benchmark.hh](#)

4.2 Dokumentacja klasy Framework

Modeluje interfejs programu.

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

Diagram dziedziczenia dla Framework



Metody publiczne

- virtual void [WczytajDane](#) (const char *nazwaPliku, unsigned int n)=0
Wczytanie danych z pliku.
- virtual void [Start](#) (const unsigned int k)=0
Wykonanie części obliczeniowej programu.
- virtual void [Zwolnij](#) ()=0
Zwalnia pamięć po teście.
- virtual void [Pokaz](#) ()=0

4.2.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanych w ramach kursu.

Definicja w linii 24 pliku Framework.hh.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 `virtual void Framework::Pokaz () [pure virtual]`

Implementowany w [ListArr2x< typ >](#).

4.2.2.2 virtual void Framework::Start (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.
----	----------	--

Implementowany w [ListArr2x< typ >](#) i [InterfejsADT< typ >](#).

4.2.2.3 virtual void Framework::WczytajDane (const char * *nazwaPliku*, 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 >](#) i [InterfejsADT< typ >](#).

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

Zwalnia pamięć zajmowaną przez obiekty wykorzystane do testów

Implementowany w [ListArr2x< typ >](#) i [InterfejsADT< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

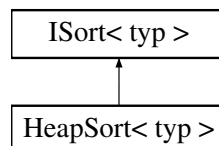
- [Framework.hh](#)

4.3 Dokumentacja szablonu klasy HeapSort< typ >

Heap Sort.

```
#include <HeapSort.hh>
```

Diagram dziedziczenia dla HeapSort< typ >



Metody publiczne

- void [Sort](#) (const int lewy, const int prawy, [Iterable](#)< typ > &tab)
Sortowanie przez kopcowanie.

Metody prywatne

- void [Kopcuje](#) (const int rozmiarKopca, const int i, [Iterable](#)< typ > &tab)
Porównuje el. kopca.
- void [BudujKopiec](#) (const int rozmiar, [Iterable](#)< typ > &tab)
Tworzy kopiec.

4.3.1 Opis szczegółowy

```
template<class typ>class HeapSort< typ >
```

Plik zawiera definicję klasy [HeapSort](#)

[HeapSort](#)

Klasa modelująca obiekt potrafiący wykonać sortowanie przez kopcowanie na kontenerze [Iterable](#).

Definicja w linii 19 pliku HeapSort.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 `template<class typ> void HeapSort< typ >::BudujKopiec (const int rozmiar, Iterable< typ > & tab)`
`[inline], [private]`

Tworzy kopiec z tablicy o podanym rozmiarze

Parametry

in	<i>rozmiar</i>	- rozmiar tablicy
in	<i>tab</i>	- referencja do sortowanego kontenera

Definicja w linii 57 pliku HeapSort.hh.

4.3.2.2 `template<class typ> void HeapSort< typ >::Kopcuje (const int rozmiarKopca, const int i, Iterable< typ > & tab)`
`[inline], [private]`

Porównuje i ustawia elementy kopca w odpowiedniej kolejności

Parametry

in	<i>rozmiarKopca</i>	- rozmiar kopca który sortujemy
in	<i>i</i>	- numer gałęzi kopca
in	<i>tab</i>	- referencja do sortowanego kontenera

Definicja w linii 31 pliku HeapSort.hh.

4.3.2.3 `template<class typ> void HeapSort< typ >::Sort (const int lewy, const int prawy, Iterable< typ > & tab)`
`[inline], [virtual]`

Realizuje algorytm sortowania przez kopcowanie. Algorytm sortuje tablicę od początku - pierwszy argument jest ignorowany.

Parametry

in	<i>lewy</i>	- indeks pierwszego elementu do posortowania
in	<i>prawy</i>	- indeks ostatniego elementu do posortowania
in	<i>tab</i>	- referencja do kontenera

Implementuje [ISort< typ >](#).

Definicja w linii 76 pliku HeapSort.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

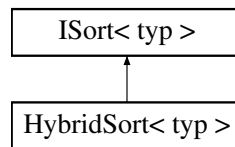
- [HeapSort.hh](#)

4.4 Dokumentacja szablonu klasy HybridSort< typ >

Definicja klasy [HybridSort](#).

```
#include <HybridSort.hh>
```

Diagram dziedziczenia dla HybridSort< typ >



Metody publiczne

- void [Sort](#) (const int lewy, const int prawy, [Iterable](#)< typ > &tab)
Sortowanie hybrydowe.

4.4.1 Opis szczegółowy

`template<class typ>class HybridSort< typ >`

Plik zawiera definicję klasy [HybridSort](#)

The [HybridSort](#) class

Klasa modeluje obiekt potrafiący sortować kontener typu [Iterable](#), algorytm wykorzystuje zoptymalizowany (mediana z trzech algorytm sortowania szybkiego oraz algorytm sortowania przez wstawianie

Definicja w linii 22 pliku HybridSort.hh.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 `template<class typ> void HybridSort< typ >::Sort (const int lewy, const int prawy, Iterable< typ > & tab)`
`[inline], [virtual]`

Metoda realizuje algorytm sortowania hybrydowego bazujący na zoptymalizowanym ze względu na wybór pivota (mediana z trzech) algorytmowi Sortowania Szybkiego oraz jako algorytm pomocniczy wykorzystane zostało sortowanie przez wstawianie.

Parametry

in	<i>lewy</i>	- indeks pierwszego elementu z listy do posortowania
in	<i>prawy</i>	- indeks ostatniego elementu z listy do posortowania
in	<i>tab</i>	- referencja do sortowanego kontenera

Implementuje [ISort](#)< typ >.

Definicja w linii 39 pliku HybridSort.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

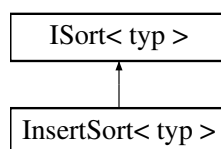
- [HybridSort.hh](#)

4.5 Dokumentacja szablonu klasy InsertSort< typ >

Insert Sort.

```
#include <InsertSort.hh>
```

Diagram dziedziczenia dla InsertSort< typ >



Metody publiczne

- void [Sort](#) (const int lewy, const int prawy, [Iterable](#)< typ > &tab)
Sortowanie przez wstawianie.

4.5.1 Opis szczegółowy

`template<class typ>class InsertSort< typ >`

Plik zawiera definicję klasy [InsertSort](#)

[InsertSort](#)

Klasa modelująca obiekt potrafiący wykonać sortowanie przez wstawianie na kontenerze [Iterable](#).

Definicja w linii 19 pliku InsertSort.hh.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 `template<class typ> void InsertSort< typ >::Sort (const int lewy, const int prawy, Iterable< typ > & tab)`
[inline], [virtual]

Metoda realizuje algorytm sortowania przez wstawianie.

Parametry

in	<i>lewy</i>	- indeks pierwszego elementu do posortowania
in	<i>prawy</i>	- indeks ostatniego elementu do posortowania
in	<i>tab</i>	- referencja do kontenera

Implementuje [ISort](#)< typ >.

Definicja w linii 33 pliku InsertSort.hh.

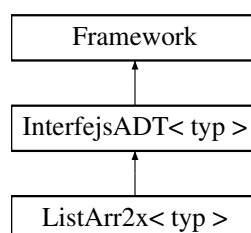
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [InsertSort.hh](#)

4.6 Dokumentacja szablonu klasy InterfejsADT< typ >

`#include <InterfejsADT.hh>`

Diagram dziedziczenia dla InterfejsADT< typ >



Metody publiczne

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

4.6.1 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.

4.6.2 Dokumentacja funkcji składowych

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

Pobiera element z typu danych

Parametry

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

Implementowany w `ListArr2x< typ >`.

4.6.2.2 `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>	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola na które chcemy dodać element

Implementowany w `ListArr2x< typ >`.

4.6.2.3 `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 >`.

4.6.2.4 `template<class typ> void InterfejsADT< typ >::Start (const unsigned int k) [pure virtual]`

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

Parametry

<code>in</code>	<code>k</code>	- ilość elementów dla których mają zostać wykonane obliczenia.
-----------------	----------------	--

Implementuje [Framework](#).

Implementowany w [ListArr2x< typ >](#).

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

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

Parametry

<code>in</code>	<code>nazwaPliku</code>	- nazwa pliku z danymi
<code>in</code>	<code>n</code>	- ilość danych do wczytania

Implementuje [Framework](#).

Implementowany w [ListArr2x< typ >](#).

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

Zwalnia pamięć zajmowaną przez daną strukturę

Implementuje [Framework](#).

Implementowany w [ListArr2x< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

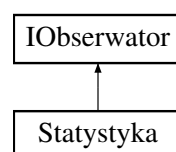
- [InterfejsADT.hh](#)

4.7 Dokumentacja klasy IObserwator

Klasa [IObserwator](#).

```
#include <IObserwator.hh>
```

Diagram dziedziczenia dla IObserwator



Metody publiczne

- `virtual void Aktualizuj ()=0`
Aktualizuj.

4.7.1 Opis szczegółowy

Plik zawiera definicję klasy IObserwator.

The [IObserwator](#) class

Klasa modeluje interfejs obiektu będącego obserwatorem.

Definicja w linii 17 pliku IObserwator.hh.

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 `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:

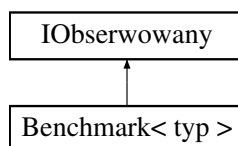
- [IObserwator.hh](#)

4.8 Dokumentacja klasy IObservowany

Interfejs obserwowanego.

```
#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.

4.8.1 Opis szczegółowy

W pliku zawarta jest definicja interfejsu obserwowanego

The [IObservowany](#) class

Klasa czysto wirtualna modelująca interfejs obiektu obserwowanego.

Definicja w linii 19 pliku IObservowany.hh.

4.8.2 Dokumentacja funkcji składowych

4.8.2.1 `virtual void IObservowany::DodajObserwatora (IObserwator * nowyObserwator) [pure virtual]`

Dodaje nowego obserwatora do listy obserwatorów danego obiektu.

Parametry

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

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

4.8.2.2 `virtual void IObszerowany::PowiadomObserwatorow () [pure virtual]`

Powiadamia obserwatorów o wydarzeniu.

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

4.8.2.3 `virtual void IObszerowany::UsunObserwatora (IObszerwator * 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:

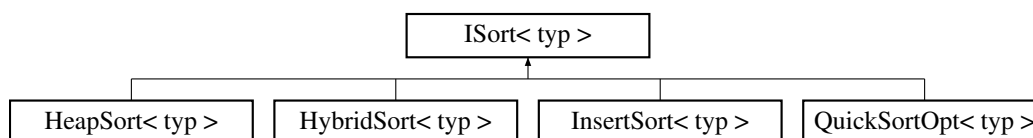
- [IObszerowany.hh](#)

4.9 Dokumentacja szablonu klasy ISort< typ >

Interfejs [ISort](#).

```
#include <ISort.hh>
```

Diagram dziedziczenia dla ISort< typ >



Metody publiczne

- `virtual void Sort (const int lewy, const int prawy, Iterable< typ > &tab)=0`
Sort.

4.9.1 Opis szczegółowy

```
template<class typ>class ISort< typ >
```

Plik zawiera definicję interfejsu ISortable

Interfejs [ISort](#)

Klasa modeluje pojęcie interfejsu [ISort](#), który tworzy interfejs użytkownika dla algorytmów sortujących.

Definicja w linii 21 pliku ISort.hh.

4.9.2 Dokumentacja funkcji składowych

4.9.2.1 `template<class typ> virtual void ISort< typ >::Sort (const int lewy, const int prawy, Iterable< typ > & tab)`
`[pure virtual]`

Metoda wykonująca algorytm sortowania na kontenerze podanym w argumencie.

Parametry

in	<i>tab</i>	- referencja do sortowanego kontenera
in	<i>lewy</i>	- początkowy indeks sortowania
in	<i>prawy</i>	- końcowy indeks sortowania

Implementowany w [QuickSortOpt< typ >](#), [HeapSort< typ >](#), [HybridSort< typ >](#) i [InsertSort< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

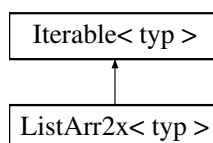
- [ISort.hh](#)

4.10 Dokumentacja szablonu klasy Iterable< typ >

Interfejs [Iterable](#).

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

Diagram dziedziczenia dla Iterable< typ >

**Metody publiczne**

- virtual typ [operator\[\]](#) (unsigned int indeks) const =0
 [operator \[\]](#)
- virtual void [Zamien](#) (int a, int b)=0

4.10.1 Opis szczegółowy

```
template<class typ>class Iterable< typ >
```

Plik zawiera definicję interfejsu [Iterable](#)

Interfejs [Iterable](#)

Klasa modeluje pojęcie interfejsu [Iterable](#), który umożliwia wgląd w kontener, oraz udostępnia interfejsowi ISortable metodę umożliwiającą kolejkovanie elementów.

Definicja w linii 22 pliku Iterable.hh.

4.10.2 Dokumentacja funkcji składowych

4.10.2.1 `template<class typ> virtual typ Iterable< typ >::operator[] (unsigned int indeks) const` [pure virtual]

Przeciążenie opratora[] w celu umożliwienia przeglądania zawrtości kontenera. Indeksujemy od 0.

Parametry

in	<i>indeks</i>	- indeks elementu, którego wartość ma zostać odczytana
----	---------------	--

Zwracane wartości

<i>zwraca</i>	wartość znajdującą się na danym indeksie kontenera
---------------	--

Implementowany w [ListArr2x< typ >](#).

4.10.2.2 `template<class typ> virtual void Iterable< typ >::Zamien (int a, int b) [pure virtual]`

Implementowany w [ListArr2x< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

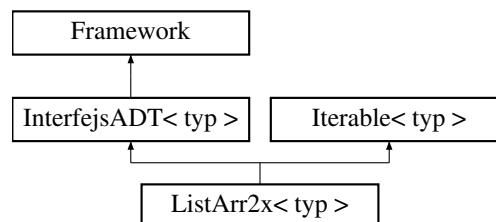
- [Iterable.hh](#)

4.11 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 ListyArr2x.
- void [pop](#) (const unsigned int pole)
Pobiera element z ListyArr2x.
- unsigned int [size](#) () const
Wielkość listy.
- void [Start](#) (const unsigned int k)
Metoda do testów.
- void [WczytajDane](#) (const char *nazwaPliku, unsigned int n)
Wczytuje dane z pliku.
- void [Zwolnij](#) ()
Zwalnia pamięć
- void [Pokaz](#) ()
Wyświetla elementy Listy.
- typ [operator](#)[] (unsigned int indeks) const
operator []
- void [Zamien](#) (int a, int b)
Zamienia elementy listy.

Metody prywatne

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

Atrybuty prywatne

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

4.11.1 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 25 pliku ListArr2x.hh.

4.11.2 Dokumentacja konstruktora i destruktor

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

Konstruktor alokujący tablicę jednoelementową z której będzie tworzona lista

Definicja w linii 98 pliku ListArr2x.hh.

4.11.3 Dokumentacja funkcji składowych

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

Dodaje daną do listy na określony indeks

Parametry

<i>dana</i>	- wartość która ma zostać umieszczona na liście
<i>pole</i>	- indeks pola na którym ma zostać umieszczona wartość

Definicja w linii 77 pliku ListArr2x.hh.

4.11.3.2 `template<class typ> typ ListArr2x< typ >::operator[] (unsigned int indeks) const [inline], [virtual]`

Przeciążenie opratora[] w celu umożliwienia przeglądania zawrości listy. Indeksujemy od 0.

Parametry

<i>in</i>	<i>indeks</i>	- indeks elementu, którego wartość ma zostać odczytana
-----------	---------------	--

Zwracane wartości

<i>zwraca</i>	wartość znajdującą się na danym indeksie listy
---------------	--

Implementuje [Iterable< typ >](#).

Definicja w linii 254 pliku ListArr2x.hh.

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

Metoda wypsuje na terminal elementy znajdujące się na liście

Implementuje [Framework](#).

Definicja w linii 238 pliku ListArr2x.hh.

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

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

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 150 pliku ListArr2x.hh.

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

Dodaje nowy element do ListyArr2x

Parametry

<i>in</i>	<i>dana</i>	- element który chcemy umieścić na liście
<i>in</i>	<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 115 pliku ListArr2x.hh.

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

Informuje o ilości elementów znajdujących się na LiścieArr2x

Zwracane wartości

-	zwraca liczbę elementów ListyArr2x
---	------------------------------------

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 184 pliku ListArr2x.hh.

4.11.3.7 `template<class typ> void ListArr2x< typ >::Start (const unsigned int k) [inline],[virtual]`

Metoda, której czas wykonania ma zostać zmierzony

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 194 pliku ListArr2x.hh.

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

Usuwa z listy element o podanym indeksie

Parametry

in	pole	- indeks elementu do usunięcia.
----	------	---------------------------------

Definicja w linii 59 pliku ListArr2x.hh.

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

Wczytuje dane z pliku do [ListArr2x](#)

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 207 pliku ListArr2x.hh.

4.11.3.10 `template<class typ> void ListArr2x< typ >::Zamien (int a, int b)` `[inline], [virtual]`

Zamienia dwa elementy listy o polach podanych w wywołaniu

Parametry

in	a	- indeks pierwszego elementu do zamiany
in	b	- indeks drugiego elementu do zamiany

Implementuje [Iterable< typ >](#).

Definicja w linii 266 pliku ListArr2x.hh.

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

Zwalnia pamięć zaalokowaną przez [ListArr2x](#)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 224 pliku ListArr2x.hh.

4.11.4 Dokumentacja atrybutów składowych

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

Aktualny rozmiar ListyArr2x

Definicja w linii 49 pliku ListArr2x.hh.

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

Aktualny rozmiar tablicy.

Definicja w linii 41 pliku ListArr2x.hh.

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

Wskaźnik na dynamiczną tablicę tworzącą ListęArr2x

Definicja w linii 33 pliku ListArr2x.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

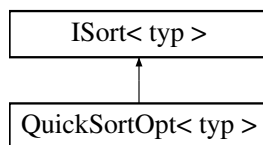
- [ListArr2x.hh](#)

4.12 Dokumentacja szablonu klasy QuickSortOpt< typ >

Definicja klasy [QuickSortOpt](#).

```
#include <QuickSortOpt.hh>
```

Diagram dziedziczenia dla QuickSortOpt< typ >



Metody publiczne

- void [Sort](#) (const int lewy, const int prawy, [Iterable](#)< typ > &tab)
Zoptymalizowane Szybkie Sortowanie.

Metody prywatne

- int [Partition](#) (int lewy, int prawy, [Iterable](#)< typ > &tab)
- int [MedianaTrzech](#) (const int a, const int b, const int c, [Iterable](#)< typ > &tab) const
Znajduje mediane.

Przyjaciele

- template<class U >
class [HybridSort](#)

4.12.1 Opis szczegółowy

```
template<class typ>class QuickSortOpt< typ >
```

Plik zawiera definicję klasy [QuickSortOpt](#)

The [QuickSortOpt](#) class

Klasa modeluje obiekt potrafiący sortować kontener typu [Iterable](#), algorytmem szybkiego syrtowania zoptymalizowanym o wybór pivota na podstawie mediany z trzech.

Definicja w linii 22 pliku QuickSortOpt.hh.

4.12.2 Dokumentacja funkcji składowych

4.12.2.1 `template<class typ> int QuickSortOpt< typ >::MedianaTrzech (const int a, const int b, const int c, Iterable< typ > & tab) const` `[inline], [private]`

Znajduje mediane wartości z trzech podanych elementów kontenera

Parametry

in	a	- indeks pierwszego elementu do liczenia mediany
----	---	--

in	<i>b</i>	- indeks drugiego elementu do liczenia mediany
in	<i>c</i>	- indeks trzeciego elementu do liczenia mediany
in	<i>tab</i>	- referencja do sortowanego kontenera

Zwracane wartości

-	zwraca indeks elementu będącego medianą z trzech wartości podanych elementów
---	--

Definicja w linii 65 pliku QuickSortOpt.hh.

4.12.2.2 `template<class typ> int QuickSortOpt< typ >::Partition (int lewy, int prawy, Iterable< typ > & tab)`
`[inline], [private]`

Partycjonowanie kontenera

Metoda będąca częścią algorytmu Sortowania Szybkiego. Dzieli przekazany fragment kontenera na dwie części - lewy z elementami mniejszymi od wybranego pivota i prawa z elementami większymi od wybranego pivota. Pivot jest dobierany za pomocą liczenia mediany z trzech elementów: pierwszego, środkowego i ostatniego.

Parametry

in	<i>lewy</i>	- indeks pierwszego elementu z kontenera do posortowania
in	<i>prawy</i>	- indeks ostatniego elementu z kontenera do posortowania
in	<i>tab</i>	- referencja do sortowanego kontenera

Definicja w linii 38 pliku QuickSortOpt.hh.

4.12.2.3 `template<class typ> void QuickSortOpt< typ >::Sort (const int lewy, const int prawy, Iterable< typ > & tab)`
`[inline], [virtual]`

Realizuje zoptymalizowany ze względu na wybór pivota algorytm szybkiego sortowania elementów konteneru

Parametry

in	<i>lewy</i>	- indeks pierwszego elementu tworzącego kontener do posortowania
in	<i>prawy</i>	- indeks ostatniego elementu tworzącego kontener do posortowania
in	<i>tab</i>	- referencja do sortowanego kontenera

Implementuje `ISort< typ >`.

Definicja w linii 94 pliku QuickSortOpt.hh.

4.12.3 Dokumentacja przyjaciół i funkcji związanych

4.12.3.1 `template<class typ> template<class U > friend class HybridSort` `[friend]`

Definicja w linii 78 pliku QuickSortOpt.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

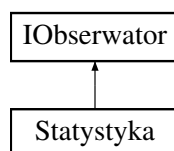
- [QuickSortOpt.hh](#)

4.13 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.

4.13.1 Opis szczegółowy

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

Definicja w linii 27 pliku Statystyka.hh.

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 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 prob.

Definicja w linii 12 pliku Statystyka.cpp.

4.13.2.2 Statystyka::~Statystyka () [inline]

Zwalnia pamiec zaalokowana na dynamiczne tablice przechowujace statystyke.

Definicja w linii 108 pliku Statystyka.hh.

4.13.3 Dokumentacja funkcji składowych

4.13.3.1 void Statystyka::Aktualizuj () [virtual]

Aktualizuje pozyskiwane dane dotyczace wynikow testu: Jezeli stoper nie odlicza to uruchamia odliczanie, Jezeli stoper odlicza to go zatrzymuje i sumuje czasy powtorzen. Gdy nastapi wykonanie wszystkich pomiarow w probie to uzupełnia talice przechowujaca srednie czasy kazdej proby.

Implementuje [IObserwator](#).

Definicja w linii 44 pliku Statystyka.cpp.

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

Zapisuje statystyke do pliku o nazwie podanej w argumencie. Plik zapisany zostaje w sposob, gdzie kazda nowa linia wyglada nastepujaco: 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.

4.13.4 Dokumentacja atrybutów składowych

4.13.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze srednimi czasami wykonania kolejnych prob.

Definicja w linii 51 pliku Statystyka.hh.

4.13.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 35 pliku Statystyka.hh.

4.13.4.3 unsigned int Statystyka::IloscPowtorzen [private]

Przechowuje ilosc wykonywanych powtorzen pojedynczego testu.

Definicja w linii 65 pliku Statystyka.hh.

4.13.4.4 unsigned int Statystyka::LicznikPowtorzen [private]

Zlicza ilosc wykonanych powtorzen w danej probie.

Definicja w linii 72 pliku Statystyka.hh.

4.13.4.5 unsigned int Statystyka::LicznikProb [private]

Zlicza ilość prób wykonanych prób.

Definicja w linii 79 pliku Statystyka.hh.

4.13.4.6 Stoper* Statystyka::MojStoper [private]

[Stoper](#) wykorzystywany do pomiaru czasu.

Definicja w linii 86 pliku Statystyka.hh.

4.13.4.7 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 43 pliku Statystyka.hh.

4.13.4.8 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](#)

4.14 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.

4.14.1 Opis szczegółowy

Plik zawiera definicję klasy [Stoper](#).

The [Stoper](#) class

Klasa modeluje stoper niezbędny do odliczania czasu.

Definicja w linii 20 pliku Stoper.hh.

4.14.2 Dokumentacja konstruktora i destruktora

4.14.2.1 Stoper::Stoper ()

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

Definicja w linii 3 pliku Stoper.cpp.

4.14.3 Dokumentacja funkcji składowych

4.14.3.1 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.

4.14.3.2 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.

4.14.3.3 void Stoper::Reset ()

Resetuje stoper.

Definicja w linii 19 pliku Stoper.cpp.

4.14.3.4 void Stoper::Start ()

Uruchamia odliczanie czasu.

Definicja w linii 9 pliku Stoper.cpp.

4.14.3.5 void Stoper::Stop ()

Zatrzymuje odliczanie czasu.

Definicja w linii 14 pliku Stoper.cpp.

4.14.4 Dokumentacja atrybutów składowych

4.14.4.1 `double Stoper::CzasKoncowy` `[private]`

Czas w którym odliczanie czasu zostało zatrzymane.

Definicja w linii 34 pliku `Stoper.hh`.

4.14.4.2 `double Stoper::CzasPocatkowy` `[private]`

Czas w którym stoper zaczął odliczać.

Definicja w linii 27 pliku `Stoper.hh`.

4.14.4.3 `bool Stoper::CzyLiczy` `[private]`

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

Definicja w linii 42 pliku `Stoper.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

5 Dokumentacja plików

5.1 Dokumentacja pliku `Benchmark.hh`

Definicja klasy [Benchmark](#).

```
#include "InterfejsADT.hh"
#include <ctime>
#include "Statystyka.hh"
#include "IObserwowany.hh"
#include <list>
#include "../inc/ISort.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>
```

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 HeapSort.hh

```
#include "ISort.hh"
```

Komponenty

- class [HeapSort< typ >](#)
Heap Sort.

5.4 Dokumentacja pliku HybridSort.hh

```
#include "QuickSortOpt.hh"  
#include "InsertSort.hh"
```

Komponenty

- class [HybridSort< typ >](#)
Definicja klasy [HybridSort](#).

5.5 Dokumentacja pliku InsertSort.hh

```
#include "ISort.hh"
```

Komponenty

- class [InsertSort< typ >](#)
Insert Sort.

5.6 Dokumentacja pliku InterfejsADT.hh

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

Komponenty

- class [InterfejsADT< typ >](#)

5.7 Dokumentacja pliku IObserwator.hh

Komponenty

- class `IObserwator`
Klasa IObserwator.

5.8 Dokumentacja pliku IObserwowany.hh

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

Komponenty

- class `IObserwowany`
Interfejs obserwowanego.

Definicje

- #define `IOBSERWOWANY_HH`

5.8.1 Dokumentacja definicji

5.8.1.1 #define IOBSERWOWANY_HH

Definicja w linii 2 pliku IObserwowany.hh.

5.9 Dokumentacja pliku ISort.hh

```
#include "ListArr2x.hh"
```

Komponenty

- class `ISort< typ >`
Interfejs ISort.

5.10 Dokumentacja pliku Iterable.hh

```
#include <iostream>
```

Komponenty

- class `Iterable< typ >`
Interfejs Iterable.

Definicje

- #define `ITREABLE_HH`

5.10.1 Dokumentacja definicji

5.10.1.1 #define ITREABLE_HH

Definicja w linii 2 pliku Iterable.hh.

5.11 Dokumentacja pliku ListArr2x.hh

Definicja klasy [ListArr2x](#).

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

Komponenty

- class [ListArr2x](#)< typ >
Modeluje pojęcie Listy (array)

5.11.1 Opis szczegółowy

Plik zawiera definicję klasy [ListArr2x](#) ujętej w szablon typu wraz z jej składowymi metodami.

Definicja w pliku [ListArr2x.hh](#).

5.12 Dokumentacja pliku main.cpp

Moduł główny programu.

```
#include "../inc/ListArr2x.hh"
#include "../inc/Statystyka.hh"
#include "../inc/Benchmark.hh"
#include "../inc/Pliki.hh"
#include "../inc/InsertSort.hh"
#include "../inc/HeapSort.hh"
#include "../inc/HybridSort.hh"
```

Funkcje

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

Zmienne

- const int [ILOSC_POWTORZEN](#) = 50
Ilość powtórzeń danej próby.
- const int [ILOSC_PROB](#) = 9
Ilość prób.
- const std::string [NAZWA_PLIKU_Z_DANYMI](#) = "dane.dat"

5.12.1 Opis szczegółowy

Program wykonuje serię 50 pomiarów czasu wykonania metody start dla różnych wielkości problemu obliczeniowego. OBSŁUGA PROGRAMU: Aby wywołać program należy w linii poleceń wywołać jego nazę np: "./a.out"

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

5.12.2 Dokumentacja funkcji

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

Definicja w linii 40 pliku main.cpp.

5.12.3 Dokumentacja zmiennych

5.12.3.1 `const int ILOSC_POWTORZEN = 50`

Ilość powtórzeń danej próby

Definicja w linii 28 pliku main.cpp.

5.12.3.2 `const int ILOSC_PROB = 9`

Ilość prób = ilość rozmiarów prób

Definicja w linii 36 pliku main.cpp.

5.12.3.3 `const std::string NAZWA_PLIKU_Z_DANYMI = "dane.dat"`

Definicja w linii 38 pliku main.cpp.

5.13 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.

5.13.1 Opis szczegółowy

Plik zawiera definicje funkcji związanych z obsługą plików.

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

5.13.2 Dokumentacja funkcji

5.13.2.1 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.

5.13.2.2 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 ktory chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.13.2.3 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 ktory chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.14 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.

5.14.1 Opis szczegółowy

Plik zawiera deklaracje funkcji związanych z obsługą plików

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

5.14.2 Dokumentacja funkcji

5.14.2.1 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.

5.14.2.2 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 ktory chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.14.2.3 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 ktory chcemy otworzyc
in	<i>plik</i>	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.15 Dokumentacja pliku QuickSortOpt.hh

```
#include "ISort.hh"
#include "HybridSort.hh"
```

Komponenty

- class [QuickSortOpt< typ >](#)

Definicja klasy [QuickSortOpt](#).

5.16 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy [Statystyka](#).

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

5.16.1 Opis szczegółowy

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

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

5.17 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy [Statystyka](#).

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

Komponenty

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

5.17.1 Opis szczegółowy

Zawiera definicję klasy [Statystyka](#)

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

5.18 Dokumentacja pliku Stoper.cpp

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

5.19 Dokumentacja pliku Stoper.hh

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

Komponenty

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

Skorowidz

~Statystyka
Statystyka, [25](#)

Aktualizuj
IObservator, [14](#)
Statystyka, [25](#)

Benchmark
Benchmark, [4](#)
DodajObservatora, [5](#)
IleDanych, [5](#)
IlePowtorzen, [5](#)
IleProb, [6](#)
ListaObservatorow, [6](#)
PowiadomObservatorow, [5](#)
Test, [5](#)
UsunObservatora, [5](#)

Benchmark< typ >, [3](#)
Benchmark.hh, [28](#)
BudujKopiec
HeapSort, [8](#)

Czas
Statystyka, [25](#)

CzasKoncowy
Stoper, [27](#)

CzasPoczątkowy
Stoper, [28](#)

CzyLiczy
Stoper, [28](#)

CzyOdmierza
Stoper, [27](#)

DajPomiar
Stoper, [27](#)

DodajDoListy
ListArr2x, [18](#)

DodajObservatora
Benchmark, [5](#)
IObservowany, [14](#)

Framework, [6](#)
Pokaz, [6](#)
Start, [6](#)
WczytajDane, [7](#)
Zwolnij, [7](#)

Framework.hh, [28](#)

HeapSort
BudujKopiec, [8](#)
Kopcuje, [8](#)
Sort, [8](#)

HeapSort< typ >, [7](#)

HeapSort.hh, [29](#)

HybridSort
QuickSortOpt, [23](#)
Sort, [9](#)

HybridSort< typ >, [8](#)
HybridSort.hh, [29](#)

ILOSC_POWTORZEN
main.cpp, [32](#)

ILOSC_PROB
main.cpp, [32](#)

IOBSERWOWANY_HH
IObservowany.hh, [30](#)

IObservator, [13](#)
Aktualizuj, [14](#)

IObservator.hh, [30](#)

IObservowany, [14](#)
DodajObservatora, [14](#)
PowiadomObservatorow, [14](#)
UsunObservatora, [15](#)

IObservowany.hh, [30](#)
IOBSERWOWANY_HH, [30](#)

ISort
Sort, [15](#)

ISort< typ >, [15](#)

ISort.hh, [30](#)
ITREABLE_HH
Iterable.hh, [31](#)

IleDanych
Benchmark, [5](#)

IlePowtorzen
Benchmark, [5](#)

IleProb
Benchmark, [6](#)
Statystyka, [25](#)

IloscPowtorzen
Statystyka, [25](#)

InsertSort
Sort, [10](#)

InsertSort< typ >, [9](#)

InsertSort.hh, [29](#)

InterfejsADT
pop, [11](#)
push, [11](#)
size, [11](#)
Start, [11](#)
WczytajDane, [13](#)
Zwolnij, [13](#)

InterfejsADT< typ >, [10](#)

InterfejsADT.hh, [29](#)

Iterable
Zamien, [17](#)
Iterable< typ >, [16](#)
Iterable.hh, [30](#)
ITREABLE_HH, [31](#)

Kopcuje
HeapSort, [8](#)

LicznikPowtorzen

- Statystyka, 25
- LicznikProb
 - Statystyka, 25
- ListArr2x
 - DodajDoListy, 18
 - ListArr2x, 18
 - ListArr2x, 18
 - Pokaz, 19
 - pop, 19
 - push, 19
 - RozmiarL, 21
 - RozmiarT, 21
 - size, 19
 - Start, 19
 - tab, 21
 - UsunZListy, 19
 - WczytajDane, 21
 - Zamien, 21
 - Zwolnij, 21
- ListArr2x< typ >, 17
- ListArr2x.hh, 31
- ListaObserwatorow
 - Benchmark, 6
- LosujIntDoPliku
 - Pliki.cpp, 32
 - Pliki.hh, 34
- main
 - main.cpp, 32
- main.cpp, 31
 - ILOSC_POWTORZEN, 32
 - ILOSC_PROB, 32
 - main, 32
- MedianaTrzech
 - QuickSortOpt, 22
- MojStoper
 - Statystyka, 26
- OtworzPlikIn
 - Pliki.cpp, 34
 - Pliki.hh, 35
- OtworzPlikOut
 - Pliki.cpp, 34
 - Pliki.hh, 35
- Partition
 - QuickSortOpt, 23
- Pliki.cpp, 32
 - LosujIntDoPliku, 32
 - OtworzPlikIn, 34
 - OtworzPlikOut, 34
- Pliki.hh, 34
 - LosujIntDoPliku, 34
 - OtworzPlikIn, 35
 - OtworzPlikOut, 35
- Pokaz
 - Framework, 6
 - ListArr2x, 19
- pop
 - InterfejsADT, 11
 - ListArr2x, 19
- PowiadomObserwatorow
 - Benchmark, 5
 - IObserwowany, 14
- Proba
 - Statystyka, 26
- push
 - InterfejsADT, 11
 - ListArr2x, 19
- QuickSortOpt
 - HybridSort, 23
 - MedianaTrzech, 22
 - Partition, 23
 - Sort, 23
- QuickSortOpt< typ >, 22
- QuickSortOpt.hh, 35
- Reset
 - Stoper, 27
- RozmiarL
 - ListArr2x, 21
- RozmiarT
 - ListArr2x, 21
- size
 - InterfejsADT, 11
 - ListArr2x, 19
- Sort
 - HeapSort, 8
 - HybridSort, 9
 - InsertSort, 10
 - ISort, 15
 - QuickSortOpt, 23
- Start
 - Framework, 6
 - InterfejsADT, 11
 - ListArr2x, 19
 - Stoper, 27
- Statystyka, 23
 - ~Statystyka, 25
 - Aktualizuj, 25
 - Czas, 25
 - IleProb, 25
 - IloscPowtorzen, 25
 - LicznikPowtorzen, 25
 - LicznikProb, 25
 - MojStoper, 26
 - Proba, 26
 - Statystyka, 24
 - SumaCzasuProby, 26
 - ZapiszStaty, 25
- Statystyka.cpp, 35
- Statystyka.hh, 35
- Stop
 - Stoper, 27
- Stoper, 26
 - CzasKoncowy, 27

- CzasPoczątkowy, [28](#)
- CzyLiczy, [28](#)
- CzyOdmierza, [27](#)
- DajPomiar, [27](#)
- Reset, [27](#)
- Start, [27](#)
- Stop, [27](#)
- Stoper, [27](#)
- Stoper.cpp, [36](#)
- Stoper.hh, [36](#)
- SumaCzasuProby
 - Statystyka, [26](#)
- tab
 - ListArr2x, [21](#)
- Test
 - Benchmark, [5](#)
- UsunObserwatora
 - Benchmark, [5](#)
 - IObservowany, [15](#)
- UsunZListy
 - ListArr2x, [19](#)
- WczytajDane
 - Framework, [7](#)
 - InterfejsADT, [13](#)
 - ListArr2x, [21](#)
- Zamien
 - Iterable, [17](#)
 - ListArr2x, [21](#)
- ZapiszStaty
 - Statystyka, [25](#)
- Zwolnij
 - Framework, [7](#)
 - InterfejsADT, [13](#)
 - ListArr2x, [21](#)