

PAMSI lab VI

Generated by Doxygen 1.8.6

Fri Apr 24 2015 16:19:26

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	Benchmark< typ > Class Template Reference	7
4.1.1	Detailed Description	7
4.1.2	Constructor & Destructor Documentation	7
4.1.2.1	Benchmark	7
4.1.3	Member Function Documentation	9
4.1.3.1	Test	9
4.1.4	Member Data Documentation	9
4.1.4.1	IleDanych	9
4.1.4.2	IlePowtorzen	9
4.1.4.3	IleProb	9
4.1.4.4	stat	9
4.2	Kolejka< typ >::Element Struct Reference	9
4.2.1	Detailed Description	10
4.2.2	Constructor & Destructor Documentation	10
4.2.2.1	Element	10
4.2.3	Member Data Documentation	10
4.2.3.1	nastepny	10
4.2.3.2	wartosc	10
4.3	Lista< typ >::Element Struct Reference	10
4.3.1	Detailed Description	11
4.3.2	Constructor & Destructor Documentation	11
4.3.2.1	Element	11

4.3.3	Member Data Documentation	11
4.3.3.1	nastepny	11
4.3.3.2	wartosc	11
4.4	Stos< typ >::Element Struct Reference	11
4.4.1	Detailed Description	12
4.4.2	Constructor & Destructor Documentation	12
4.4.2.1	Element	12
4.4.3	Member Data Documentation	12
4.4.3.1	nastepny	12
4.4.3.2	wartosc	12
4.5	Framework Class Reference	12
4.5.1	Detailed Description	13
4.5.2	Member Function Documentation	13
4.5.2.1	StartMsort	13
4.5.2.2	WczytajDane	13
4.5.2.3	Zwolnij	14
4.6	InterfejsADT< typ > Class Template Reference	14
4.6.1	Detailed Description	14
4.6.2	Member Function Documentation	15
4.6.2.1	pop	15
4.6.2.2	push	15
4.6.2.3	size	15
4.6.2.4	StartMsort	15
4.6.2.5	WczytajDane	15
4.6.2.6	Zwolnij	17
4.7	Kolejka< typ > Class Template Reference	17
4.7.1	Detailed Description	18
4.7.2	Constructor & Destructor Documentation	18
4.7.2.1	Kolejka	18
4.7.3	Member Function Documentation	18
4.7.3.1	pop	18
4.7.3.2	push	18
4.7.3.3	size	19
4.7.3.4	Start	19
4.7.3.5	WczytajDane	19
4.7.3.6	Zwolnij	19
4.7.4	Member Data Documentation	19
4.7.4.1	Koniec	19
4.7.4.2	Poczatek	20
4.7.4.3	Rozmiar	20

4.8	Lista< typ > Class Template Reference	20
4.8.1	Detailed Description	21
4.8.2	Constructor & Destructor Documentation	21
4.8.2.1	Lista	21
4.8.3	Member Function Documentation	21
4.8.3.1	operator[]	21
4.8.3.2	pop	21
4.8.3.3	push	22
4.8.3.4	size	22
4.8.3.5	Start	22
4.8.3.6	WczytajDane	22
4.8.3.7	Zwolnij	23
4.8.4	Member Data Documentation	23
4.8.4.1	Koniec	23
4.8.4.2	Początek	23
4.8.4.3	Rozmiar	23
4.9	ListArr1< typ > Class Template Reference	23
4.9.1	Detailed Description	24
4.9.2	Constructor & Destructor Documentation	24
4.9.2.1	ListArr1	24
4.9.3	Member Function Documentation	24
4.9.3.1	pop	24
4.9.3.2	push	24
4.9.3.3	size	25
4.9.3.4	Start	25
4.9.3.5	WczytajDane	25
4.9.3.6	Zwolnij	25
4.9.4	Member Data Documentation	25
4.9.4.1	RozmiarL	25
4.9.4.2	RozmiarT	25
4.9.4.3	tab	26
4.10	ListArr2x< typ > Class Template Reference	26
4.10.1	Detailed Description	27
4.10.2	Constructor & Destructor Documentation	27
4.10.2.1	ListArr2x	27
4.10.3	Member Function Documentation	28
4.10.3.1	BudujKopiec	28
4.10.3.2	Kopiec	29
4.10.3.3	Mediana	29
4.10.3.4	Merge	29

4.10.3.5	Partycjowanie	29
4.10.3.6	Pokaz	30
4.10.3.7	pop	30
4.10.3.8	push	30
4.10.3.9	Qsort	30
4.10.3.10	QsortOpt	30
4.10.3.11	size	30
4.10.3.12	Sortowanie_Hybrydowe	31
4.10.3.13	SortowanieKopiec	31
4.10.3.14	Start	31
4.10.3.15	StartMsort	31
4.10.3.16	WczytajDane	31
4.10.3.17	Wstaw_Sort	32
4.10.3.18	Zamien	32
4.10.3.19	Zwolnij	32
4.10.4	Member Data Documentation	32
4.10.4.1	RozmiarL	32
4.10.4.2	RozmiarT	32
4.10.4.3	tab	32
4.11	Statystyka Class Reference	32
4.11.1	Detailed Description	33
4.11.2	Constructor & Destructor Documentation	33
4.11.2.1	Statystyka	33
4.11.2.2	~Statystyka	33
4.11.3	Member Function Documentation	34
4.11.3.1	operator[]	34
4.11.3.2	ZapiszStaty	35
4.11.4	Member Data Documentation	35
4.11.4.1	Czas	35
4.11.4.2	IleProb	35
4.11.4.3	Proba	35
4.12	Stos< typ > Class Template Reference	35
4.12.1	Detailed Description	36
4.12.2	Constructor & Destructor Documentation	36
4.12.2.1	Stos	36
4.12.3	Member Function Documentation	36
4.12.3.1	pop	36
4.12.3.2	push	37
4.12.3.3	size	37
4.12.3.4	Start	37

4.12.3.5	WczytajDane	37
4.12.3.6	Zwolnij	37
4.12.4	Member Data Documentation	38
4.12.4.1	Początek	38
4.12.4.2	Rozmiar	38
5	File Documentation	39
5.1	/home/bartolomeo/209296/prj/inc/Benchmark.hh File Reference	39
5.1.1	Detailed Description	39
5.2	/home/bartolomeo/209296/prj/inc/Framework.hh File Reference	39
5.2.1	Detailed Description	39
5.3	/home/bartolomeo/209296/prj/inc/Kolejka.hh File Reference	40
5.3.1	Detailed Description	40
5.4	/home/bartolomeo/209296/prj/inc/Lista.hh File Reference	40
5.4.1	Detailed Description	40
5.5	/home/bartolomeo/209296/prj/inc/ListArr1.hh File Reference	40
5.5.1	Detailed Description	41
5.6	/home/bartolomeo/209296/prj/inc/ListArr2x.hh File Reference	41
5.6.1	Detailed Description	41
5.7	/home/bartolomeo/209296/prj/inc/Pliki.hh File Reference	41
5.7.1	Detailed Description	42
5.7.2	Function Documentation	42
5.7.2.1	LosujIntDoPliku	42
5.7.2.2	OtworzPlikIn	42
5.7.2.3	OtworzPlikOut	42
5.8	/home/bartolomeo/209296/prj/inc/Statystyka.hh File Reference	42
5.8.1	Detailed Description	43
5.9	/home/bartolomeo/209296/prj/inc/Stos.hh File Reference	43
5.9.1	Detailed Description	43
5.10	/home/bartolomeo/209296/prj/src/main.cpp File Reference	43
5.10.1	Detailed Description	44
5.10.2	Macro Definition Documentation	44
5.10.2.1	ILOSC_POWTORZEN	44
5.10.2.2	ILOSC_PROB	44
5.11	/home/bartolomeo/209296/prj/src/Pliki.cpp File Reference	44
5.11.1	Detailed Description	44
5.11.2	Function Documentation	44
5.11.2.1	LosujIntDoPliku	44
5.11.2.2	OtworzPlikIn	45
5.11.2.3	OtworzPlikOut	45

5.12 /home/bartolomeo/209296/prj/src/Statystyka.cpp File Reference	45
5.12.1 Detailed Description	45
Index	46

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Benchmark< typ >	7
Kolejka< typ >::Element	9
Lista< typ >::Element	10
Stos< typ >::Element	11
Framework	12
InterfejsADT< typ >	14
Kolejka< typ >	17
Lista< typ >	20
ListArr1< typ >	23
ListArr2x< typ >	26
Stos< typ >	35
Statystyka	32

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Benchmark< typ >	
Modeluje pojęcie Benchmarku	7
Kolejka< typ >::Element	
Modeluje jeden element Kolejki	9
Lista< typ >::Element	
Modeluje jeden element Listy	10
Stos< typ >::Element	
Modeluje jeden element Stosu	11
Framework	
Modeluje interfejs programu	12
InterfejsADT< typ >	14
Kolejka< typ >	
Modeluje pojęcie Kolejki	17
Lista< typ >	
Modeluje pojęcie listy	20
ListArr1< typ >	
Modeluje pojęcie Listy (array)	23
ListArr2x< typ >	
Modeluje pojęcie Listy (array)	26
Statystyka	
Modeluje pojęcie statystyki	32
Stos< typ >	
Modeluje pojęcie Stosu	35

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

/home/bartolomeo/209296/prj/inc/Benchmark.hh	
Definicja klasy Benchmark	39
/home/bartolomeo/209296/prj/inc/Framework.hh	
Definicja klasy Framework	39
/home/bartolomeo/209296/prj/inc/InterfejsADT.hh	??
/home/bartolomeo/209296/prj/inc/Kolejka.hh	
Definicja klasy Kolejka	40
/home/bartolomeo/209296/prj/inc/Lista.hh	
Eefinicja klasy Lista	40
/home/bartolomeo/209296/prj/inc/ListArr1.hh	
Definicja klasy ListaArr1	40
/home/bartolomeo/209296/prj/inc/ListArr2x.hh	
Definicja klasy ListArr1	41
/home/bartolomeo/209296/prj/inc/Pliki.hh	
Funkcje obsługi plików	41
/home/bartolomeo/209296/prj/inc/Statystyka.hh	
Zawiera definicję klasy Statystyka	42
/home/bartolomeo/209296/prj/inc/Stos.hh	
Zawiera definicję Stosu	43
/home/bartolomeo/209296/prj/src/main.cpp	
Moduł główny programu	43
/home/bartolomeo/209296/prj/src/Pliki.cpp	
Definicje funkcji obsługi plików	44
/home/bartolomeo/209296/prj/src/Statystyka.cpp	
Zawiera definicję metod klasy Statystyka	45

Chapter 4

Class Documentation

4.1 Benchmark< typ > Class Template Reference

Modeluje pojęcie Benchmarku.

```
#include <Benchmark.hh>
```

Public Member Functions

- [Benchmark](#) (const unsigned int ileProb, unsigned int *ileDanych, unsigned int ilePowtorzen)
Konstruktor 2 argumentowy.
- void [Test](#) ([Framework](#) *f, std::string nazwaPliku)
Testowanie algorytmu.

Private Attributes

- [Statystyka](#) * [stat](#)
Statystyki testu.
- unsigned int [ileProb](#)
Ilość prób.
- unsigned int * [ileDanych](#)
Tablica liczności serii.
- unsigned int [ilePowtorzen](#)
Ilość powtórzeń

4.1.1 Detailed Description

```
template<class typ>class Benchmark< typ >
```

Modeluje pojęcie Benchmarku.

Modeluje pojęcie Benchmarku czyli obiektu mierzącego czas wykonywania algoytmu

4.1.2 Constructor & Destructor Documentation

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

Konstruktor 2 argumentowy.

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

Parameters

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

4.1.3 Member Function Documentation

4.1.3.1 `template<class typ> void Benchmark< typ >::Test (Framework * I, std::string nazwaPliku) [inline]`

Testowanie algorytmu.

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

Parameters

in	<i>I</i>	- obiekt klasy na której zostanie przeprowadzony test
in	<i>nazwaPliku</i>	- nazwa pliku do którego zostaną zapisane statystyki

4.1.4 Member Data Documentation

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

Tablica licznosci serii.

Tablica z licznosciami elementów dla kolejnych serii

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

Ilość powtórzeń

Ilość powtórzeń każdej serii

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

Ilość prób.

Ilość powtórzeń każdej serii

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

Statystyki testu.

Pole przechowuje wyniki testów

The documentation for this class was generated from the following file:

- </home/bartolomeo/209296/prj/inc/Benchmark.hh>

4.2 Kolejka< typ >::Element Struct Reference

Modeluje jeden element Kolejki.

Public Member Functions

- [Element](#) (typ *k*)

Konstruktor daną przekazywaną w argumencie.

Public Attributes

- typ [wartosc](#)

Wartosc Elementu.

- [Element](#) * [nastepny](#)

Wskaźnik na kolejny [Element](#) Kolejki.

4.2.1 Detailed Description

```
template<class typ>struct Kolejka< typ >::Element
```

Modeluje jeden element Kolejki.

Modeluje jeden nierozłączny element Kolejki - przechowywaną daną oraz wskaźnik na następny element;

4.2.2 Constructor & Destructor Documentation

4.2.2.1 `template<class typ > Kolejka< typ >::Element::Element (typ k)` `[inline]`

Konstruktor daną przekazywaną w argumencie.

Konstruktor zapisujący w Elemencie na końcu Kolejki daną podaną w argumencie i ustawiający wskaźnik na NULL

Parameters

<i>in</i>	<i>k</i>	- daną która ma zostać dodana na koniec Kolejki
-----------	----------	---

4.2.3 Member Data Documentation

4.2.3.1 `template<class typ > Element* Kolejka< typ >::Element::nastepny`

Wskaźnik na kolejny [Element](#) Kolejki.

Wskaźnik na kolejny [Element](#) Kolejki

4.2.3.2 `template<class typ > typ Kolejka< typ >::Element::wartosc`

Wartosc Elementu.

Wartość Elementu - przechowywanej wartości przez dany [Element](#) Kolejki

The documentation for this struct was generated from the following file:

- `/home/bartolomeo/209296/prj/inc/Kolejka.hh`

4.3 Lista< typ >::Element Struct Reference

Modeluje jeden element Listy.

Public Member Functions

- [Element](#) (typ k)

Konstruktor daną przekazywaną w argumencie.

Public Attributes

- typ [wartosc](#)

Wartosc Elementu.

- [Element](#) * [nastepny](#)

Wskaźnik na kolejny [Element](#) Listy.

4.3.1 Detailed Description

```
template<class typ>struct Lista< typ >::Element
```

Modeluje jeden element Listy.

Modeluje jeden nierozłączny element listy - przechowywaną daną oraz wskaźnik na następny element;

4.3.2 Constructor & Destructor Documentation

4.3.2.1 `template<class typ > Lista< typ >::Element::Element (typ k)` `[inline]`

Konstruktor daną przekazywaną w argumencie.

Konstruktor zapisujący w Elemencie na końcu Listy daną podaną w argumencie i ustawiający wskaźnik na NULL

Parameters

<code>in</code>	<code>k</code>	- daną która ma zostać dodana na koniec Listy
-----------------	----------------	---

4.3.3 Member Data Documentation

4.3.3.1 `template<class typ > Element* Lista< typ >::Element::nastepny`

Wskaźnik na kolejny [Element](#) Listy.

Wskaźnik na kolejny [Element](#) Listy

4.3.3.2 `template<class typ > typ Lista< typ >::Element::wartosc`

Wartosc Elementu.

Wartość Elementu - przechowywanej wartości przez dany [Element](#) listy

The documentation for this struct was generated from the following file:

- `/home/bartolomeo/209296/prj/inc/Lista.hh`

4.4 Stos< typ >::Element Struct Reference

Modeluje jeden element Stosu.

Public Member Functions

- [Element](#) (typ k)
Konstruktor daną przekazywaną w argumencie.

Public Attributes

- typ [wartosc](#)
Wartosc Elementu.
- [Element](#) * [nastepny](#)
Wskaźnik na kolejny [Element](#) Stosu.

4.4.1 Detailed Description

```
template<class typ>struct Stos< typ >::Element
```

Modeluje jeden element Stosu.

Modeluje jeden nierozłączny element Stosu - przechowywaną daną oraz wskaźnik na następny element;

4.4.2 Constructor & Destructor Documentation

4.4.2.1 `template<class typ > Stos< typ >::Element::Element (typ k)` `[inline]`

Konstruktor daną przekazywaną w argumencie.

Konstruktor zapisujący w Elemencie na końcu Listy daną podaną w argumencie i ustawiający wskaźnik na NULL

Parameters

<code>in</code>	<code>k</code>	- dana która ma zostać dodana na koniec Stosu
-----------------	----------------	---

4.4.3 Member Data Documentation

4.4.3.1 `template<class typ > Element* Stos< typ >::Element::nastepny`

Wskaźnik na kolejny [Element](#) Stosu.

Wskaźnik na kolejny [Element](#) Stosu

4.4.3.2 `template<class typ > typ Stos< typ >::Element::wartosc`

Wartosc Elementu.

Wartość Elementu - przechowywanej wartości przez dany [Element](#) Stosu

The documentation for this struct was generated from the following file:

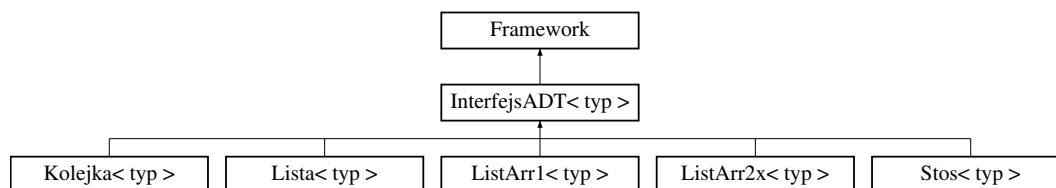
- `/home/bartolomeo/209296/prj/inc/Stos.hh`

4.5 Framework Class Reference

Modeluje interfejs programu.

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

Inheritance diagram for Framework:



Public Member Functions

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

4.5.1 Detailed Description

Modeluje interfejs programu.

Modeluje interfejs do programów wykonywanych w ramach kursu.

4.5.2 Member Function Documentation

4.5.2.1 virtual void Framework::StartMsort (unsigned int k) [pure virtual]

Wykonanie części obliczeniowej programu.

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

Parameters

in	k	- ilość elementów dla których mają zostać wykonane obliczenia.
----	---	--

Implemented in [ListArr2x< typ >](#), and [InterfejsADT< typ >](#).

4.5.2.2 virtual void Framework::WczytajDane (const std::string nazwaPliku, unsigned int n) [pure virtual]

Wczytanie danych z pliku.

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

Parameters

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

Implemented in [ListArr2x< typ >](#), and [InterfejsADT< typ >](#).

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

Zwalnia pamięć po teście.

Zwalnia pamięć zajmowaną przez obiekty wykorzystane do testów

Implemented in [ListArr2x< typ >](#), [ListArr1< typ >](#), [Kolejka< typ >](#), [Lista< typ >](#), [Stos< typ >](#), and [InterfejsADT< typ >](#).

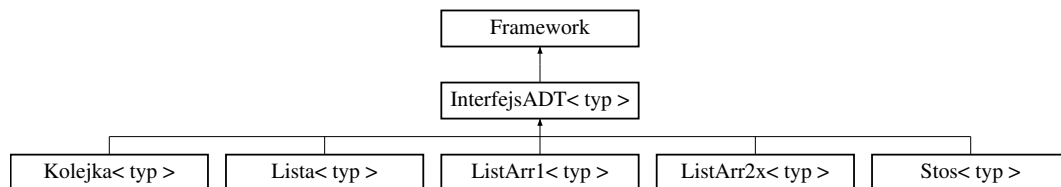
The documentation for this class was generated from the following file:

- [/home/bartolomeo/209296/prj/inc/Framework.hh](#)

4.6 InterfejsADT< typ > Class Template Reference

```
#include <InterfejsADT.hh>
```

Inheritance diagram for InterfejsADT< typ >:



Public Member Functions

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

4.6.1 Detailed Description

```
template<class typ>class InterfejsADT< typ >
```

\ brief Definiuje interfejs użytkownika

Definiuje interfejs użytkownika dla listy, stosu i kolejki.

4.6.2 Member Function Documentation

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

Pobiera element.

Pobiera element z typu danych

Parameters

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

Return values

<i>zwraca</i>	wartość danego elementu
---------------	-------------------------

Implemented in [ListArr2x< typ >](#), [Lista< typ >](#), [Kolejka< typ >](#), [Stos< typ >](#), and [ListArr1< typ >](#).

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

Dodaje kolejny element.

Dodaje kolejny element do typu danych

Parameters

<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

Implemented in [ListArr2x< typ >](#), [Kolejka< typ >](#), [Lista< typ >](#), [Stos< typ >](#), and [ListArr1< typ >](#).

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

Liczność elementów.

Informuje o liczności elementów obecnie przechowywanych

Return values

<i>zwraca</i>	ilość przechowywanych elementów
---------------	---------------------------------

Implemented in [ListArr2x< typ >](#), [Lista< typ >](#), [Kolejka< typ >](#), [Stos< typ >](#), and [ListArr1< typ >](#).

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

Wykonanie części obliczeniowej programu.

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

Parameters

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

Implements [Framework](#).

Implemented in [ListArr2x< typ >](#).

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

Wczytanie danych z pliku.

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

Parameters

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

Implements [Framework](#).

Implemented in [ListArr2x< typ >](#).

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

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez daną strukturę

Implements [Framework](#).

Implemented in [ListArr2x< typ >](#), [ListArr1< typ >](#), [Kolejka< typ >](#), [Lista< typ >](#), and [Stos< typ >](#).

The documentation for this class was generated from the following file:

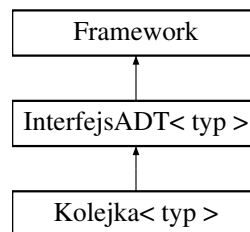
- /home/bartolomeo/209296/prj/inc/InterfejsADT.hh

4.7 Kolejka< typ > Class Template Reference

Modeluje pojęcie Kolejki.

```
#include <Kolejka.hh>
```

Inheritance diagram for Kolejka< typ >:



Classes

- struct [Element](#)
Modeluje jeden element Kolejki.

Public Member Functions

- [Kolejka](#) ()
Konstruktor pustej Kolejki.
- void [Zwolnij](#) ()
Destruktor Kolejki.
- void [push](#) (typ dana, unsigned int pole=0)
Dodaje daną do Kolejki.
- void [pop](#) (unsigned int pole=0)
Usuwa element z Kolejki.
- unsigned int [size](#) ()
Sprawdza rozmiar Kolejki.

- void [WczytajDane](#) (const char *nazwaPliku, unsigned int n)
Wczytuje dane z pliku.
- void [Start](#) (const unsigned int k)
Proces obliczeniowy.

Private Attributes

- [Element](#) * [Poczatek](#)
Wskaźnik na pierwszy element Kolejki.
- [Element](#) * [Koniec](#)
Wskaźnik na ostatni element Kolejki.
- unsigned int [Rozmiar](#)
Aktualny rozmiar Kolejki.

4.7.1 Detailed Description

template<class typ>class Kolejka< typ >

Modeluje pojęcie Kolejki.

Modeluje pojęcie Kolejki zadeklarowanego w szablonie typu Uwaga! Kolejkę indeksujemy od 0.

4.7.2 Constructor & Destructor Documentation

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

Konstruktor pustej Kolejki.

Konstruktor bezargumentowy pustej Kolejki tworzy obiekt z wskaźnikiem początek pokazującym na NULL.

4.7.3 Member Function Documentation

4.7.3.1 template<class typ > void Kolejka< typ >::pop (unsigned int *pole* = 0) [inline],[virtual]

Usuwa element z Kolejki.

Usuwa pierwszy element z Kolejki UWAGA! Nie zmieniać drugiego argumentu wywołania, bądź ustawoć 0!

Parameters

in	<i>pole</i>	- numer elementu w Kolejce którzy wyrzucimy, domyślnie 0, zmiana podczas wywołania nie ma wpływu na działanie metody;
----	-------------	---

Implements [InterfejsADT< typ >](#).

4.7.3.2 template<class typ > void Kolejka< typ >::push (typ *dana*, unsigned int *pole* = 0) [inline],[virtual]

Dodaje daną do Kolejki.

Dodaje daną podaną jako pierwszy argument wywołania na koniec Kolejki Uwaga! nie zmieniać drugiego argumentu wywołania!

Parameters

in	<i>dana</i>	- dana którą chcemy dodać do Kolejki
in	<i>pole</i>	- numer miejsca gdzie zostanie dodany element - domyślnie koniec kolejki, zmiana argumentu podczas wywołania nie wpływa na działanie metody.

Implements [InterfejsADT< typ >](#).

4.7.3.3 `template<class typ> unsigned int Kolejka< typ >::size () [inline],[virtual]`

Sprawdza rozmiar Kolejki.

Sprawdza ile aktualnie elementów znajduje się w Kolejce

Return values

<i>zwraca</i>	ilość elementów znajdujących się aktualnie w Kolejce
---------------	--

Implements [InterfejsADT< typ >](#).

4.7.3.4 `template<class typ> void Kolejka< typ >::Start (const unsigned int k) [inline]`

Proces obliczeniowy.

Wykonuje proces obliczeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypadku tworzy Kolejke k elementową wypełnioną stałą liczbą '3'.

Parameters

in	<i>k</i>	- ilość danych dla których ma zostać przeprowadzona procedura obliczeniowa
----	----------	--

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

Wczytuje dane z pliku.

Wczytuje dane zamieszczone w pliku do Kolejki. Każdą nową daną umieszcza na końcu Kolejki.

Parameters

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

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

Destruktor Kolejki.

Zwalnia zaalokowana przez Kolejke pamiec

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez Kolejke

Implements [InterfejsADT< typ >](#).

4.7.4 Member Data Documentation

4.7.4.1 `template<class typ> Element* Kolejka< typ >::Koniec [private]`

Wskaźnik na ostatni element Kolejki.

Wskaźnik na ostatni element kolejki zwiększający szybkość dodawania danych na końcu

4.7.4.2 `template<class typ > Element* Kolejka< typ >::Poczatek` `[private]`

Wskaźnik na pierwszy element Kolejki.

Wskaźnik na pierwszy element Kolejki

4.7.4.3 `template<class typ > unsigned int Kolejka< typ >::Rozmiar` `[private]`

Aktualny rozmiar Kolejki.

Przechowuje aktualną ilość Elementów znajdujących się w Kolejce

The documentation for this class was generated from the following file:

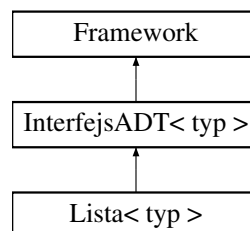
- </home/bartolomeo/209296/prj/inc/Kolejka.hh>

4.8 Lista< typ > Class Template Reference

Modeluje pojęcie listy.

```
#include <Lista.hh>
```

Inheritance diagram for Lista< typ >:



Classes

- struct [Element](#)
Modeluje jeden element Listy.

Public Member Functions

- [Lista](#) ()
Konstruktor puste listy.
- void [Zwolnij](#) ()
Destruktor listy.
- void [push](#) (typ dana, unsigned int pole)
Dodaje daną do Listy.
- typ [pop](#) (unsigned int pole)
Usuwa element z Listy.
- unsigned int [size](#) ()
Sprawdza rozmiar Listy.
- void [WczytajDane](#) (const char *nazwaPliku, unsigned int n=0)
Wczytuje dane z pliku.

- typ `operator[]` (size_t pole) const
Wyciąga wartość elementu Listy.
- void `Start` (const unsigned int k)
Proces obliczeniowy.

Private Attributes

- Element * `Początek`
Wskaźnik na pierwszy element Listy.
- Element * `Koniec`
Wskaźnik na ostatni element listy.
- unsigned int `Rozmiar`
Aktualny rozmiar Listy.

4.8.1 Detailed Description

`template<class typ>class Lista< typ >`

Modeluje pojęcie listy.

Modeluje pojęcie listy zadeklarowanego w szablonie typu Uwaga! Listę indeksujemy od 0.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 `template<class typ > Lista< typ >::Lista () [inline]`

Konstruktor puste listy.

Konstruktor bezargumentowy pustej listy tworzy obiekt z wskaźnikiem początek pokazującym na NULL.

4.8.3 Member Function Documentation

4.8.3.1 `template<class typ > typ Lista< typ >::operator[] (size_t pole) const [inline]`

Wyciąga wartość elementu Listy.

Wyluskuje wartość danego elementu z Listy

Parameters

<code>in</code>	<code>pole</code>	- "indeks" z którego chcemy pobrać wartość indeksujemy od 0!
-----------------	-------------------	--

Return values

-	zwraca wartość elementu z danego pola lub '-1' w przypadku błędu
---	--

4.8.3.2 `template<class typ > typ Lista< typ >::pop (unsigned int pole) [inline], [virtual]`

Usuwa element z Listy.

Usuwa interesujący nas element z Listy. Jeżeli chcesz usunąć pierwszy element wywołaj pole nr '0'. Dla ostatniego elementu wywołaj pole nr '`Lista.size()-1`'.

Parameters

<i>in</i>	<i>pole</i>	- numer elementu Listy z którego chcemy pobrać daną
-----------	-------------	---

Return values

<i>zwraca</i>	wartość danego elementu listy lub '-1' w przypadku błędu
---------------	--

Implements [InterfejsADT< typ >](#).

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

Dodaje daną do Listy.

Dodaje daną podaną jako pierwszy argument wywołania na określone drugim argumentem miejsce w Liście

Parameters

<i>in</i>	<i>dana</i>	- dana którą chcemy dodać do listy
<i>in</i>	<i>pole</i>	- numer elementu listy na który chcemy dodać daną (szieze() jeżeli na koniec)

Implements [InterfejsADT< typ >](#).

4.8.3.4 `template<class typ > unsigned int Lista< typ >::size ()` `[inline], [virtual]`

Sprawdza rozmiar Listy.

Sprawdza ile aktualnie elementów znajduje się na Liście

Return values

<i>zwraca</i>	ilość elementów znajdujących się aktualnie na liście
---------------	--

Implements [InterfejsADT< typ >](#).

4.8.3.5 `template<class typ > void Lista< typ >::Start (const unsigned int k)` `[inline]`

Proces obliczeniowy.

Wykonuje proces obliczeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypadku tworzy Listę k elementową wypełnioną stałą liczbą '3'.

Parameters

<i>in</i>	<i>k</i>	- ilość danych dla których ma zostać przeprowadzona procedura obliczeniowa
-----------	----------	--

4.8.3.6 `template<class typ > void Lista< typ >::WczytajDane (const char * nazwaPliku, unsigned int n = 0)` `[inline]`

Wczytuje dane z pliku.

Wczytuje dane zamieszczone w pliku do Listy. Każdą nową daną umieszcza na końcu listy.

Parameters

<i>in</i>	<i>nazwaPliku</i>	- nazwa pliku z danymi
<i>in</i>	<i>n</i>	- ilość danych do wczytania (domyślnie 0 - wszystkie dane z pliku, zmiana wartości nie ma wpływu na działanie metody w aktualnej wersji)

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

Destruktor listy.

Zwalnia zaalokowana przez liste pamiec

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez listę

Implements [InterfejsADT< typ >](#).

4.8.4 Member Data Documentation

4.8.4.1 `template<class typ> Element* Lista< typ >::Koniec [private]`

Wskaźnik na ostatni element listy.

Wskaźnik na ostatni element listy

4.8.4.2 `template<class typ> Element* Lista< typ >::Początek [private]`

Wskaźnik na pierwszy element Listy.

Wskaźnik na pierwszy element Listy

4.8.4.3 `template<class typ> unsigned int Lista< typ >::Rozmiar [private]`

Aktualny rozmiar Listy.

Przechowuje aktualną ilość Elementów znajdujących się na Liście

The documentation for this class was generated from the following file:

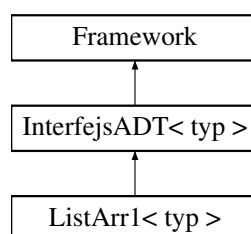
- [/home/bartolomeo/209296/prj/inc/Lista.hh](#)

4.9 ListArr1< typ > Class Template Reference

Modeluje pojęcie Listy (array)

```
#include <ListArr1.hh>
```

Inheritance diagram for ListArr1< typ >:



Public Member Functions

- [ListArr1 \(\)](#)
Konstruktor bezargumentowy.
- void [push](#) (typ dana, unsigned int pole)

- Dodaje element do ListyArr1.*
- typ [pop](#) (unsigned int pole)
Pobiera element z ListyArr1.
- unsigned int [size](#) ()
Wielkość listy.
- void [Start](#) (const unsigned int k)
Metoda testująca czas.
- void [WczytajDane](#) (const char *nazwaPliku, unsigned int n)
Wczytuje dane z pliku.
- void [Zwolnij](#) ()
Zwalnia pamięć

Private Attributes

- typ * [tab](#)
Wskaźnik na dynamiczną tablicę
- unsigned int [RozmiarT](#)
Rozmiar tablicy.
- unsigned int [RozmiarL](#)
Rozmiar Listy.

4.9.1 Detailed Description

`template<class typ>class ListArr1< typ >`

Modeluje pojęcie Listy (array)

Modeluje pojęcie Listy opartej na dynamicznej tablicy. Dodając elementy zwiększa tablicę o 1.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 `template<class typ > ListArr1< typ >::ListArr1 () [inline]`

Konstruktor bezargumentowy.

Konstruktor alokujący tablicę jednoelementową z której będzie tworzona lista

4.9.3 Member Function Documentation

4.9.3.1 `template<class typ > typ ListArr1< typ >::pop (unsigned int pole) [inline],[virtual]`

Pobiera element z ListyArr1.

Pobiera element z Listy Arr1 usuwając go z niej i zmniejszając rozmiar.

param[in] - pole - nr pola z którego chcemy pobrać element

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

Implements [InterfejsADT< typ >](#).

4.9.3.2 `template<class typ > void ListArr1< typ >::push (typ dana, unsigned int pole) [inline],[virtual]`

Dodaje element do ListyArr1.

Dodaje nowy element do ListyArr1

Parameters

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()

Implements [InterfejsADT< typ >](#).

4.9.3.3 `template<class typ> unsigned int ListArr1< typ >::size () [inline],[virtual]`

Wielkość listy.

Informuje o ilości elementów znajdujących się na LiścieArr1

Return values

-	zwraca liczbę elementów ListyArr1
---	-----------------------------------

Implements [InterfejsADT< typ >](#).

4.9.3.4 `template<class typ> void ListArr1< typ >::Start (const unsigned int k) [inline]`

Metoda testująca czas.

Metoda testująca czas wczytania n elementów na ListęArr1

Parameters

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

4.9.3.5 `template<class typ> void ListArr1< typ >::WczytajDane (const char * nazwaPliku, unsigned int n) [inline]`

Wczytuje dane z pliku.

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

4.9.3.6 `template<class typ> void ListArr1< typ >::Zwolnij () [inline],[virtual]`

Zwalnia pamięć

Zwalnia pamięć zaalokowaną przez [ListArr1](#)

Implements [InterfejsADT< typ >](#).

4.9.4 Member Data Documentation

4.9.4.1 `template<class typ> unsigned int ListArr1< typ >::RozmiarL [private]`

Rozmiar Listy.

Aktualny rozmiar ListyArr1

4.9.4.2 `template<class typ> unsigned int ListArr1< typ >::RozmiarT [private]`

Rozmiar tablicy.

Aktualny rozmiar tablicy.

4.9.4.3 `template<class typ> typ* ListArr1< typ >::tab` [private]

Wkaźnik na dynamiczną tablicę

Wskaźnik na dynamiczną tablicę tworzącą ListęArr1

The documentation for this class was generated from the following file:

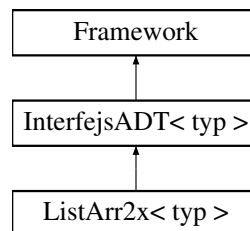
- `/home/bartolomeo/209296/prj/inc/ListArr1.hh`

4.10 ListArr2x< typ > Class Template Reference

Modeluje pojęcie Listy (array)

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

Inheritance diagram for ListArr2x< typ >:



Public Member Functions

- void `Wstaw_Sort` (typ *W, int l, int p)
Sortowanie przez Wstawianie Metoda ma za zadanie posortować tablicę przyjmowaną jako argument.
- `ListArr2x` ()
Konstruktor bezargumentowy.
- void `push` (typ dana, unsigned int pole)
Dodaje element do ListyArr1.
- typ `pop` (unsigned int pole)
Pobiera element z ListyArr1.
- unsigned int `size` ()
Wielkość listy.
- void `WczytajDane` (const std::string PlikIn, unsigned int n)
Wczytuje dane z pliku.
- void `Qsort` (int l, int h)
Metoda wykorzystująca sortowanie szybkie.
- void `QsortOpt` (int lewy, int prawy1)
Zoptymalizowane Sortowanie Szybkie.
- void `Pokaz` ()
Metoda wypisująca elementy listy.
- void `MSort` (typ *T, int p, int k)
- void `SortowanieKopiec` (int Rozmiar)
Metoda sortowania przez Kopcowanie.
- void `Sortowanie_Hybrydowe` (int l, int h)
Metoda sortowania hybrydowego.

Private Member Functions

- int [Mediana](#) (typ *W)
Mediana Metoda wyznaczająca medianę dla tablicy 3 elementowej. Jest to metoda pomocnicza, wykorzystywana przy optymalizacji doboru pivotu w sortowaniu szybkim.
- void [StartMsort](#) (unsigned int k)
Metoda testująca czas.
- void [Start](#) ()
Metoda testująca czas.
- void [Zamien](#) (typ &i, typ &j)
Metoda zamieniająca Metoda ma za zadanie zamienić miejscami elementy wybrane przez argumenty wywołania.
- int [Partycjonowanie](#) (int p, int k)
Metoda segregująca.
- void [Merge](#) (typ *Temp, int l, int s, int p)
Metoda Dzielnica tablice.
- void [Zwolnij](#) ()
Zwalnia pamięć
- void [Kopiec](#) (int Rozmiar, const int i)
Metoda składająca kopiec.
- void [BudujKopiec](#) (int Rozmiar)
Metoda tworząca kopiec.

Private Attributes

- typ * [tab](#)
Wskaźnik na dynamiczną tablicę
- unsigned int [RozmiarT](#)
Rozmiar tablicy.
- unsigned int [RozmiarL](#)
Rozmiar Listy.

4.10.1 Detailed Description

template<class typ>class ListArr2x< typ >

Modeluje pojęcie Listy (array)

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

4.10.2 Constructor & Destructor Documentation

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

Konstruktor bezargumentowy.

Konstruktor alokujący tablicę jednoelementową z której będzie tworzona lista

4.10.3 Member Function Documentation

4.10.3.1 `template<class typ > void ListArr2x< typ >::BudujKopiec (int Rozmiar)` `[inline], [private]`

Metoda tworząca kopiec.

Metoda ma za zadanie utworzyć abstrakcyjny kopiec z tablicy o podanym poprzez argument rozmiarze

Parameters

in	<i>Rozmiar</i>	- Rozmiar kopca
----	----------------	-----------------

4.10.3.2 `template<class typ > void ListArr2x< typ >::Kopiec (int Rozmiar, const int i)` [inline],[private]

Metoda składająca kopiec.

Metoda ma za zadanie poprzez porównywanie i ustawianie elementów odtworzenie porządku kopcowego.

Parameters

in	<i>Rozmiar</i>	- Rozmiar kopca
in	<i>i</i>	- indeks ostatniego elementu podzbioru

4.10.3.3 `template<class typ > int ListArr2x< typ >::Mediana (typ * W)` [inline],[private]

Mediana Metoda wyznaczająca medianę dla tablicy 3 elementowej. Jest to metoda pomocnicza, wykorzystywana przy optymalizacji doboru pivotu w sortowaniu szybkim.

Returns

Zwraca indeks na którym znajduje się mediana w tablicy wejściowej

4.10.3.4 `template<class typ > void ListArr2x< typ >::Merge (typ * Temp, int l, int s, int p)` [inline],[private]

Metoda Dzielnika tablicy.

Metoda ma za zadanie przekopiować zawartość zbioru głównego do tablicy tymczasowej. Następnie operując na kopii ustawia wskaźniki na początki kolejnych zbiorów i porównywane są wskazane wartości. Mniejsze wpisujemy do zbioru głównego i przesuwamy odpowiedni wskaźnik. Czynnosc wykonujemy rekurencyjnie aż do momentu gdy jeden ze wskaźników osiągnie koniec zbioru

Parameters

in	<i>Temp</i>	- Wskaźnik na tablicę pomocniczą
in	<i>l</i>	- Początkowy indeks tablicy
in	<i>s</i>	- Środkowy indeks tablicy
in	<i>p</i>	- Końcowy indeks tablicy

4.10.3.5 `template<class typ > int ListArr2x< typ >::Partycjonowanie (int p, int k)` [inline],[private]

Metoda segregująca.

Metoda ma za zadanie wybrać element, który ma być użyty do podziału i przenosi wszystkie elementy mniejsze na lewo od tego elementu, a większe elementy na prawo od wybranego elementu

Parameters

in	<i>p</i>	- początkowy indeks podzbioru
in	<i>k</i>	- końcowy indeks podzbioru

Returns

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

Metoda wypisująca elementy listy.

Metoda ma za zadanie wypisać wszystkie elementy znajdujące się obecnie na liście danych

Implements [InterfejsADT< typ >](#).

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

Pobiera element z ListyArr1.

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

Implements [InterfejsADT< typ >](#).

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

Dodaje element do ListyArr1.

Dodaje nowy element do ListyArr1

Parameters

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()

Implements [InterfejsADT< typ >](#).

4.10.3.9 `template<class typ > void ListArr2x< typ >::Qsort (int l, int h) [inline]`

Metoda wykorzystująca sortowanie szybkie.

Parameters

in	<i>l</i>	- początkowy indeks tablicy
in	<i>h</i>	- końcowy indeks tablicy

4.10.3.10 `template<class typ > void ListArr2x< typ >::QsortOpt (int lewy, int prawy1) [inline]`

Zoptymalizowane Sortowanie Szybkie.

Metoda modeluje algorytm sortowania szybkiego z zaimplementowanym algorytmem doboru pivotu, tak aby nie został wybrany najmniejszy element w danym podzbiore. [in] *lewy* - początkowy indeks podzbioru

Parameters

in	<i>prawy</i>	- końcowy indeks podzbioru
----	--------------	----------------------------

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

Wielkość listy.

Informuje o ilości elementów znajdujących się na LiścieArr1

Return values

-	zwraca liczbę elementów ListyArr1
---	-----------------------------------

Implements [InterfejsADT< typ >](#).

4.10.3.12 `template<class typ > void ListArr2x< typ >::Sortowanie_Hybrydowe (int l, int h) [inline]`

Metoda sortowania hybrydowego.

Metoda ta jest implementacja algorytmu sortowania hybrydowego, bedacego polaczeniem sortowania szybkiego i sortowania przez wstawianie Po zakonczeniu rekurencyjnych wywoan Partycjowania, tablica jest podzielona na szereg malych podzbiorow o o rozmiarze nie przekraczajacemu ustalonego progu. Zbioru sa porozdzielana elementami ktore wykorzystywane byly jako elementy osiowe. Dla czesciowo posortowanej tablicy wywoływane jest sortowanie przez wstawianie, ktore jest wydajne dla tablic o malych rozmiarach

Parameters

<i>in</i>	<i>l</i>	- indeks poczatkowego elementu pozbioru
<i>in</i>	<i>h</i>	- indeks koncowego elementu podzbioru

4.10.3.13 `template<class typ > void ListArr2x< typ >::SortowanieKopiec (int Rozmiar) [inline]`

Metoda sortowania przez Kopcowanie.

Metoda realizujaca sortowanie rosnace,wykorzystujac przy tym kopiec.

Parameters

<i>in</i>	<i>Rozmiar</i>	- Rozmiar kopca do zbudowania, ilość danych do posortowania.
-----------	----------------	--

4.10.3.14 `template<class typ > void ListArr2x< typ >::Start () [inline],[private],[virtual]`

Metoda testujaca czas.

Metoda testujaca czas wczytania n elementów na ListęArr1

Implements [InterfejsADT< typ >](#).

4.10.3.15 `template<class typ > void ListArr2x< typ >::StartMsort (unsigned int k) [inline],[private],[virtual]`

Metoda testujaca czas.

Metoda testujaca czas wczytania n elementów na ListęArr1

Parameters

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

Implements [InterfejsADT< typ >](#).

4.10.3.16 `template<class typ > void ListArr2x< typ >::WczytajDane (const std::string PlikIn, unsigned int n) [inline],[virtual]`

Wczytuje dane z pliku.

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

Implements [InterfejsADT< typ >](#).

4.10.3.17 `template<class typ > void ListArr2x< typ >::Wstaw_Sort (typ * W, int l, int p)` `[inline]`

Sortowanie przez Wstawianie Metoda ma za zadanie posortować tablice przyjmowaną jako argument.

Parameters

<code>in</code>	<code>T</code>	- Wskaznik na tablice z danymi wejściowymi
<code>in</code>	<code>l</code>	- Początkowy indeks elementu sortowanego podzbioru
<code>in</code>	<code>p</code>	- końcowy indeks elementu sortowanego podzbioru

4.10.3.18 `template<class typ > void ListArr2x< typ >::Zamien (typ & i, typ & j)` `[inline]`, `[private]`

Metoda zamieniająca Metoda ma za zadanie zamienić miejscami elementy wybrane przez argumenty wywołania.

Parameters

<code>in</code>	<code>i</code>	- Adres elementu podlegający zamianie
<code>in</code>	<code>j</code>	- Adres elementu podlegający zamianie

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

Zwalnia pamięć

Zwalnia pamięć zaalokowaną przez [ListArr1](#)

Implements [InterfejsADT< typ >](#).

4.10.4 Member Data Documentation

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

Rozmiar Listy.

Aktualny rozmiar ListyArr2x

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

Rozmiar tablicy.

Aktualny rozmiar tablicy.

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

Wskaźnik na dynamiczną tablicę

Wskaźnik na dynamiczną tablicę tworzącą ListęArr2x

The documentation for this class was generated from the following file:

- </home/bartolomeo/209296/prj/inc/ListArr2x.hh>

4.11 Statystyka Class Reference

Modeluje pojęcie statystyki.


```
#include <Statystyka.hh>
```

Public Member Functions

- [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.

Private Attributes

- unsigned int [IleProb](#)

Ilość prób.

- unsigned int * [Proba](#)

Tablica z rozmiarami prób.

- double * [Czas](#)

Średni czas wykonania danej próby.

4.11.1 Detailed Description

Modeluje pojęcie statystyki.

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

4.11.2 Constructor & Destructor Documentation

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

Konstruktor z dwoma parametrami.

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

Parameters

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

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

Destruktor - zwalnia pamięć

Zwalnia pamięć zaalokowaną na dynamiczne tablice przechowujące statystykę.

4.11.3 Member Function Documentation

4.11.3.1 `double& Statystyka::operator[] (unsigned int i)` `[inline]`

Indeksuje tablicę czasową

Zwraca referencję do i-tego indeksu tablicy czasowej.

Parameters

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

Return values

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

4.11.3.2 void Statystyka::ZapiszStaty (std::string nazwaPliku)

Zapisuje statystykę do pliku.

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

4.11.4 Member Data Documentation

4.11.4.1 double* Statystyka::Czas [private]

Średni czas wykonania danej próby.

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

4.11.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób.

Ilość prób do utworzenia statystyki

4.11.4.3 unsigned int* Statystyka::Proba [private]

Tablica z rozmiarami prób.

Wskaźnik na tablicę zawierającą wielkości danych prób.

The documentation for this class was generated from the following files:

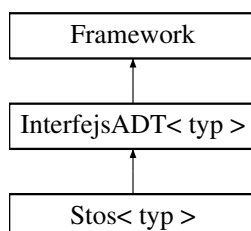
- [/home/bartolomeo/209296/prj/inc/Statystyka.hh](#)
- [/home/bartolomeo/209296/prj/src/Statystyka.cpp](#)

4.12 Stos< typ > Class Template Reference

Modeluje pojęcie Stosu.

```
#include <Stos.hh>
```

Inheritance diagram for Stos< typ >:



Classes

- struct [Element](#)
Modeluje jeden element Stosu.

Public Member Functions

- [Stos](#) ()
Konstruktor pustego Stosu.
- void [Zwolnij](#) ()
Destruktor Stosu.
- void [push](#) (typ dana, unsigned int pole=0)
Dodaje daną do Listy.
- void [pop](#) (unsigned int pole=0)
Usuwa element ze Stosu.
- unsigned int [size](#) ()
Sprawdza rozmiar Stosu.
- void [WczytajDane](#) (const char *nazwaPliku, unsigned int n)
Wczytuje dane z pliku.
- void [Start](#) (const unsigned int k)
Proces obliczeniowy.

Private Attributes

- [Element](#) * [Poczatek](#)
Wskaźnik na pierwszy element Stosu.
- unsigned int [Rozmiar](#)
Aktualny rozmiar Stosu.

4.12.1 Detailed Description

template<class typ>class Stos< typ >

Modeluje pojęcie Stosu.

Modeluje pojęcie Stosu.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 template<class typ > [Stos](#)< typ >::[Stos](#) () [inline]

Konstruktor pustego Stosu.

Konstruktor bezargumentowy pustego Stosu tworzy obiekt z wskaźnikiem początek pokazującym na NULL.

4.12.3 Member Function Documentation

4.12.3.1 template<class typ > void [Stos](#)< typ >::[pop](#) (unsigned int *pole* = 0) [inline],[virtual]

Usuwa element ze Stosu.

Usuwa 'górny' element Stosu

Parameters

<i>in</i>	<i>pole</i>	- numer elementu Listy z którego chcemy pobrać daną
-----------	-------------	---

Implements [InterfejsADT< typ >](#).

4.12.3.2 `template<class typ> void Stos< typ >::push (typ dana, unsigned int pole = 0) [inline], [virtual]`

Dodaje daną do Listy.

Dodaje daną podaną jako argument wywołania

Parameters

<i>in</i>	<i>dana</i>	- dana którą chcemy dodać do Stosu
<i>in</i>	<i>pole</i>	- numer elementu Stosu na który chcemy dodać daną, domyślnie - 0, zmiana argumentu wywołania nie ma wpływu na działanie metody

Implements [InterfejsADT< typ >](#).

4.12.3.3 `template<class typ> unsigned int Stos< typ >::size () [inline], [virtual]`

Sprawdza rozmiar Stosu.

Sprawdza ile aktualnie elementów znajduje się na Stosie

Return values

<i>zwraca</i>	ilość elementów znajdujących się aktualnie na Stosie
---------------	--

Implements [InterfejsADT< typ >](#).

4.12.3.4 `template<class typ> void Stos< typ >::Start (const unsigned int k) [inline]`

Proces obliczeniowy.

Wykonuje proces obliczeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypadku tworzy [Stos](#) k elementowy wypełniony stałą liczbą '3'.

Parameters

<i>in</i>	<i>k</i>	- ilość danych dla których ma zostać przeprowadzona procedura obliczeniowa
-----------	----------	--

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

Wczytuje dane z pliku.

Wczytuje dane zamieszczone w pliku do Stosu. Każdą nową daną umieszcza na 'górze' Stosu.

Parameters

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

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

Destruktor Stosu.

Zwalnia zaalokowaną przez [Stos](#) pamięć

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez [Stos](#)

Implements [InterfejsADT< typ >](#).

4.12.4 Member Data Documentation

4.12.4.1 `template<class typ > Element* Stos< typ >::Poczatek` `[private]`

Wskaźnik na pierwszy element Stosu.

Wskaźnik na pierwszy element Stosu

4.12.4.2 `template<class typ > unsigned int Stos< typ >::Rozmiar` `[private]`

Aktualny rozmiar Stosu.

Przechowuje aktualną ilość Elementów znajdujących się na Stosie

The documentation for this class was generated from the following file:

- `/home/bartolomeo/209296/prj/inc/Stos.hh`

Chapter 5

File Documentation

5.1 /home/bartolomeo/209296/prj/inc/Benchmark.hh File Reference

Definicja klasy [Benchmark](#).

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

Classes

- class [Benchmark](#)< typ >
Modeluje pojęcie Benchmarku.

5.1.1 Detailed Description

Definicja klasy [Benchmark](#). Plik zawiera definicję klasy [Benchmark](#) wraz z definicją jej metod.

5.2 /home/bartolomeo/209296/prj/inc/Framework.hh File Reference

Definicja klasy [Framework](#).

```
#include <iostream>
```

Classes

- class [Framework](#)
Modeluje interfejs programu.

5.2.1 Detailed Description

Definicja klasy [Framework](#). Plik zawiera definicję abstrakcyjnej klasy [Framework](#), która tworzy interfejs dla programów implementowanych podczas zajęć laboratoryjnych z PAMSI.

5.3 /home/bartolomeo/209296/prj/inc/Kolejka.hh File Reference

Definicja klasy [Kolejka](#).

```
#include "InterfejsADT.hh"
#include "Pliki.hh"
#include <ctime>
```

Classes

- class [Kolejka< typ >](#)
Modeluje pojęcie Kolejki.
- struct [Kolejka< typ >::Element](#)
Modeluje jeden element Kolejki.

5.3.1 Detailed Description

Definicja klasy [Kolejka](#). Plik zawiera definicję klasy [Kolejka](#) ujętej w szablon typu przechowywanych zmiennych więc zawiera też definicję metod klasy.

5.4 /home/bartolomeo/209296/prj/inc/Lista.hh File Reference

Eefinicja klasy [Lista](#).

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

Classes

- class [Lista< typ >](#)
Modeluje pojęcie listy.
- struct [Lista< typ >::Element](#)
Modeluje jeden element Listy.

5.4.1 Detailed Description

Eefinicja klasy [Lista](#). Plik zawiera definicję klasy lista ujętej w szablon typu przechowywanych zmiennych więc zawiera też definicję metod klasy.

5.5 /home/bartolomeo/209296/prj/inc/ListArr1.hh File Reference

Definicja klasy [ListaArr1](#).

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


Classes

- class [ListArr1](#)< typ >
Modeluje pojęcie Listy (array)

5.5.1 Detailed Description

Definicja klasy [ListArr1](#). Plik zawiera definicję klasy [ListArr1](#) ujętej w szablon typu wraz z jej składowymi metodami.

5.6 /home/bartolomeo/209296/prj/inc/ListArr2x.hh File Reference

Definicja klasy [ListArr1](#).

```
#include "InterfejsADT.hh"
#include <fstream>
#include <cstdlib>
#include <cmath>
```

Classes

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

Macros

- #define **ILE** 3
- #define **PROG** 15

5.6.1 Detailed Description

Definicja klasy [ListArr1](#). Plik zawiera definicję klasy [ListArr2x](#) ujętej w szablon typu wraz z jej składowymi metodami.

5.7 /home/bartolomeo/209296/prj/inc/Pliki.hh File Reference

Funkcje obsługi plików.

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

Functions

- 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 Detailed Description

Funkcje obsługi plików. Plik zawiera deklaracje funkcji związanych z obsługą plików

5.7.2 Function Documentation

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

Zapisuje n losowych liczb(int) do pliku.

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

Parameters

in	<i>n</i>	- ilość liczb do zapisania
in	<i>zakres</i>	- górny zakres wartości liczb

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

Otwiera plik do odczytu.

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

Parameters

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

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

Otwiera plik do zapisu czyszcząc jego zawartość

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

Parameters

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

5.8 /home/bartolomeo/209296/prj/inc/Statystyka.hh File Reference

Zawiera definicję klasy [Statystyka](#).

```
#include <iostream>
```

Classes

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

5.8.1 Detailed Description

Zawiera definicję klasy [Statystyka](#). Zawiera definicję klasy [Statystyka](#)

5.9 /home/bartolomeo/209296/prj/inc/Stos.hh File Reference

Zawiera definicję Stosu.

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

Classes

- class [Stos< typ >](#)
Modeluje pojęcie Stosu.
- struct [Stos< typ >::Element](#)
Modeluje jeden element Stosu.

5.9.1 Detailed Description

Zawiera definicję Stosu. Plik zawiera definicję klasy [Stos](#), oraz definicję jej metod, gdyż klasa ujęta jest w szablonie.

5.10 /home/bartolomeo/209296/prj/src/main.cpp File Reference

Moduł główny programu.

```
#include "../inc/Lista.hh"  
#include "../inc/Stos.hh"  
#include "../inc/Kolejka.hh"  
#include "../inc/ListArr1.hh"  
#include "../inc/ListArr2x.hh"  
#include "../inc/Statystyka.hh"  
#include "../inc/Benchmark.hh"
```

Macros

- #define [ILOSC_POWTORZEN](#) 10
Ilość powtórzeń danej próby.
- #define [ILOSC_PROB](#) 6
Ilość prób.

Functions

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

5.10.1 Detailed Description

Moduł główny programu. Program wykonuje serię 10 pomiarów czasu wykonania metody start dla różnych wielkości problemu obliczeniowego, dla każdego zaimplementowanego typu danych - LinkLista, ListaArr1, ListaArr2x. Procedura obliczeniowa polega na utworzeniu 'objektu' przechowującego n danych (stałych liczb). statystykę pomiarów zapisuje do pliku o nazwie "TypDaych.dat". gdzie "TypDanych" to odpowiednio [Lista](#), ListaArr1 i ListaArr2x

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

5.10.2 Macro Definition Documentation

5.10.2.1 #define ILOSC_POWTORZEN 10

Ilość powtórzeń danej próby.

Ilość powtórzeń danej próby

5.10.2.2 #define ILOSC_PROB 6

Ilość prób.

Ilość prób = ilość rozmiarów prób

5.11 /home/bartolomeo/209296/prj/src/Pliki.cpp File Reference

Definicje funkcji obsługi plików.

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

Functions

- 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.

5.11.1 Detailed Description

Definicje funkcji obsługi plików. Plik zawiera definicje funkcji związanych z obsługą plików.

5.11.2 Function Documentation

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

Zapisuje n losowych liczb(int) do pliku.

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

Parameters

in	<i>n</i>	- ilość liczb do zapisania
in	<i>zakres</i>	- górny zakres wartości liczb

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

Otwiera plik do odczytu.

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

Parameters

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

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

Otwiera plik do zapisu czyszac jego zawartość

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

Parameters

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

5.12 /home/bartolomeo/209296/prj/src/Statystyka.cpp File Reference

Zawiera definicję metod klasy [Statystyka](#).

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

5.12.1 Detailed Description

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

Index

~Statystyka

Statystyka, 33

/home/bartolomeo/209296/prj/inc/Benchmark.hh, 39

/home/bartolomeo/209296/prj/inc/Framework.hh, 39

/home/bartolomeo/209296/prj/inc/Kolejka.hh, 40

/home/bartolomeo/209296/prj/inc/ListArr1.hh, 40

/home/bartolomeo/209296/prj/inc/ListArr2x.hh, 41

/home/bartolomeo/209296/prj/inc/Lista.hh, 40

/home/bartolomeo/209296/prj/inc/Pliki.hh, 41

/home/bartolomeo/209296/prj/inc/Statystyka.hh, 42

/home/bartolomeo/209296/prj/inc/Stos.hh, 43

/home/bartolomeo/209296/prj/src/Pliki.cpp, 44

/home/bartolomeo/209296/prj/src/Statystyka.cpp, 45

/home/bartolomeo/209296/prj/src/main.cpp, 43

Benchmark

Benchmark, 7

IleDanych, 9

IlePowtorzen, 9

IleProb, 9

stat, 9

Test, 9

Benchmark< typ >, 7

BudujKopiec

ListArr2x, 28

Czas

Statystyka, 35

Element

Kolejka::Element, 10

Lista::Element, 11

Stos::Element, 12

Framework, 12

StartMsort, 13

WczytajDane, 13

Zwolnij, 13

ILOSC_POWTORZEN

main.cpp, 44

ILOSC_PROB

main.cpp, 44

IleDanych

Benchmark, 9

IlePowtorzen

Benchmark, 9

IleProb

Benchmark, 9

Statystyka, 35

InterfejsADT

pop, 15

push, 15

size, 15

StartMsort, 15

WczytajDane, 15

Zwolnij, 17

InterfejsADT< typ >, 14

Kolejka

Kolejka, 18

Koniec, 19

Poczatek, 20

pop, 18

push, 18

Rozmiar, 20

size, 19

Start, 19

WczytajDane, 19

Zwolnij, 19

Kolejka< typ >, 17

Kolejka< typ >::Element, 9

Kolejka::Element

Element, 10

nastepny, 10

wartosc, 10

Koniec

Kolejka, 19

Lista, 23

Kopiec

ListArr2x, 29

ListArr1

ListArr1, 24

ListArr1, 24

pop, 24

push, 24

RozmiarL, 25

RozmiarT, 25

size, 25

Start, 25

tab, 26

WczytajDane, 25

Zwolnij, 25

ListArr1< typ >, 23

ListArr2x

BudujKopiec, 28

Kopiec, 29

ListArr2x, 27

ListArr2x, 27

Mediana, 29

- Merge, [29](#)
- Partycjowanie, [29](#)
- Pokaz, [29](#)
- pop, [30](#)
- push, [30](#)
- Qsort, [30](#)
- QsortOpt, [30](#)
- RozmiarL, [32](#)
- RozmiarT, [32](#)
- size, [30](#)
- Sortowanie_Hybrydowe, [31](#)
- SortowanieKopiec, [31](#)
- Start, [31](#)
- StartMsort, [31](#)
- tab, [32](#)
- WczytajDane, [31](#)
- Wstaw_Sort, [32](#)
- Zamien, [32](#)
- Zwolnij, [32](#)
- ListArr2x< typ >, [26](#)
- Lista
 - Koniec, [23](#)
 - Lista, [21](#)
 - Poczatek, [23](#)
 - pop, [21](#)
 - push, [22](#)
 - Rozmiar, [23](#)
 - size, [22](#)
 - Start, [22](#)
 - WczytajDane, [22](#)
 - Zwolnij, [22](#)
- Lista< typ >, [20](#)
- Lista< typ >::Element, [10](#)
- Lista::Element
 - Element, [11](#)
 - nastepny, [11](#)
 - wartosc, [11](#)
- LosujIntDoPliku
 - Pliki.cpp, [44](#)
 - Pliki.hh, [42](#)
- main.cpp
 - ILOSC_POWTORZEN, [44](#)
 - ILOSC_PROB, [44](#)
- Mediana
 - ListArr2x, [29](#)
- Merge
 - ListArr2x, [29](#)
- nastepny
 - Kolejka::Element, [10](#)
 - Lista::Element, [11](#)
 - Stos::Element, [12](#)
- OtworzPlikIn
 - Pliki.cpp, [45](#)
 - Pliki.hh, [42](#)
- OtworzPlikOut
 - Pliki.cpp, [45](#)
- Pliki.hh, [42](#)
- Partycjowanie
 - ListArr2x, [29](#)
- Pliki.cpp
 - LosujIntDoPliku, [44](#)
 - OtworzPlikIn, [45](#)
 - OtworzPlikOut, [45](#)
- Pliki.hh
 - LosujIntDoPliku, [42](#)
 - OtworzPlikIn, [42](#)
 - OtworzPlikOut, [42](#)
- Poczatek
 - Kolejka, [20](#)
 - Lista, [23](#)
 - Stos, [38](#)
- Pokaz
 - ListArr2x, [29](#)
- pop
 - InterfejsADT, [15](#)
 - Kolejka, [18](#)
 - Lista, [21](#)
 - ListArr1, [24](#)
 - ListArr2x, [30](#)
 - Stos, [36](#)
- Proba
 - Statystyka, [35](#)
- push
 - InterfejsADT, [15](#)
 - Kolejka, [18](#)
 - Lista, [22](#)
 - ListArr1, [24](#)
 - ListArr2x, [30](#)
 - Stos, [37](#)
- Qsort
 - ListArr2x, [30](#)
- QsortOpt
 - ListArr2x, [30](#)
- Rozmiar
 - Kolejka, [20](#)
 - Lista, [23](#)
 - Stos, [38](#)
- RozmiarL
 - ListArr1, [25](#)
 - ListArr2x, [32](#)
- RozmiarT
 - ListArr1, [25](#)
 - ListArr2x, [32](#)
- size
 - InterfejsADT, [15](#)
 - Kolejka, [19](#)
 - Lista, [22](#)
 - ListArr1, [25](#)
 - ListArr2x, [30](#)
 - Stos, [37](#)
- Sortowanie_Hybrydowe

- ListArr2x, 31
- SortowanieKopiec
 - ListArr2x, 31
- Start
 - Kolejka, 19
 - Lista, 22
 - ListArr1, 25
 - ListArr2x, 31
 - Stos, 37
- StartMsort
 - Framework, 13
 - InterfejsADT, 15
 - ListArr2x, 31
- stat
 - Benchmark, 9
- Statystyka, 32
 - ~Statystyka, 33
 - Czas, 35
 - IleProb, 35
 - Proba, 35
 - Statystyka, 33
 - ZapiszStaty, 35
- Stos
 - Początek, 38
 - pop, 36
 - push, 37
 - Rozmiar, 38
 - size, 37
 - Start, 37
 - Stos, 36
 - WczytajDane, 37
 - Zwolnij, 37
- Stos< typ >, 35
- Stos< typ >::Element, 11
- Stos::Element
 - Element, 12
 - nastepny, 12
 - wartosc, 12
- tab
 - ListArr1, 26
 - ListArr2x, 32
- Test
 - Benchmark, 9
- wartosc
 - Kolejka::Element, 10
 - Lista::Element, 11
 - Stos::Element, 12
- WczytajDane
 - Framework, 13
 - InterfejsADT, 15
 - Kolejka, 19
 - Lista, 22
 - ListArr1, 25
 - ListArr2x, 31
 - Stos, 37
- Wstaw_Sort
 - ListArr2x, 32
- Zamien
 - ListArr2x, 32
- ZapiszStaty
 - Statystyka, 35
- Zwolnij
 - Framework, 13
 - InterfejsADT, 17
 - Kolejka, 19
 - Lista, 22
 - ListArr1, 25
 - ListArr2x, 32
 - Stos, 37