PAMSI lab VI

Generated by Doxygen 1.8.6

Fri Apr 24 2015 16:19:26

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

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 *I, 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 objektu 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 objekt klasy Benchmark i inicjuje nową statystykę dla objektu

Parameters

in	*ileProb*	- ilość prób, które zostaną wykonane
in	*ileDanych*	- wkaźnik na tablice z licznościami kolejnych serii
in	*ilePowtorzen*	- ilość powtórzeń każdej serii

4.1.3 Member Function Documentation

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

Testowanie algorytmu.

Metoda testuje algorytm w okreslonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego pez użytkownika

Parameters

in	*l*	- objekt klasy na której zostanie przeprowadzony test
in	*nazwaPliku*	- 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 liczności serii.

Tablica z licznościami elementów dla kojenych 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 seriii

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 Elemecie na końcu Kolejki daną podaną w argumencie i ustawiający wkaźnik na NULL

Parameters

in	*k*	- dana 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 Elemecie na końcu Listy daną podaną w argumencie i ustawiający wkaźnik na NULL

Parameters

in	*k*	- dana 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 Elemecie na końcu Listy daną podaną w argumencie i ustawiający wkaźnik na NULL

Parameters

in	*k*	- 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:

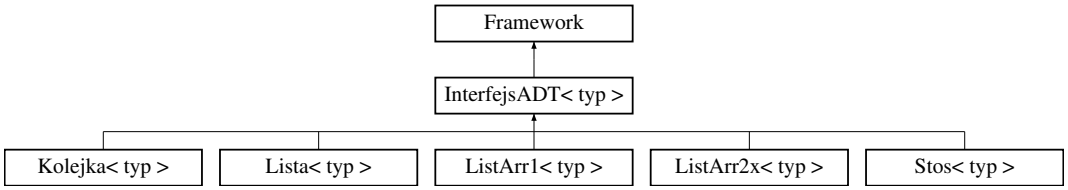
- /home/bartolomeo/209296/prj/inc/Stos.hh

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 wykonywanch 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	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- 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 objekty wykorzytsane do testów

Implemented in ListArr2x< typ >, ListArr1< typ >, Kolejka< typ >, Lista< typ >, Stos< typ >, and InterfejsADT< typ >.

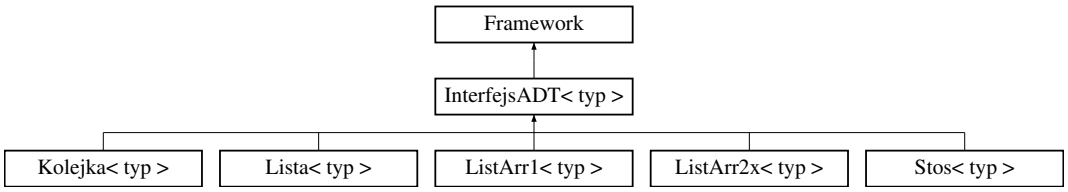
The documentation for this class was generated from the following file:

- /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 elemetów.
- 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

in	*pole*	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola z ktore pobiera element

Return values

zwraca	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

in	*dana*	- element który chcemy dorzucić do naszego typu
in	*pole*	- !!!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 elemetów.

Informuje o liczności elementów obecnie przechowywanych

Return values

zwraca	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

in	*k*	- 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	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- 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:

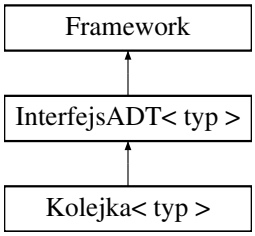
- /home/bartolomeo/209296/prj/inc/InterfejsADT.hh

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 objekt 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	*pole*	- 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	*dana*	- dana którą chcemy dodać do Kolejki
in	*pole*	- numer miejsca gdzie zostanie dodany element - domyślnie koniec koelejki, zmiana arumentu podczas wywowłania 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 znajduję się w Kolejce

Return values

zwraca	ilosć elementów znadjących 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 oblcizeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypakdu tworzy Kolejkę k elementową wypełnioną stałą liczbą '3'.

Parameters

in	*k*	- ilość danych dla których ma zostać przeprowadzona precedura obnliczeniowa

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	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- 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 Kolejkę

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 szybkowść 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 Elemenetów znajujących się w Kolejce

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

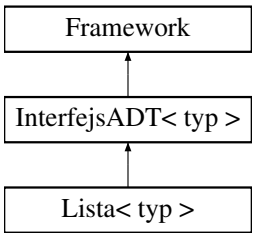
- /home/bartolomeo/209296/prj/inc/Kolejka.hh

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 * Poczatek

  Wskaźnik na pierwszy element Listy.
- Element * Koniec

  Wzkaźnik 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 objekt 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.

Wyłuskuje wartość danego elementu z Listy

Parameters

in	*pole*	- "indeks" z którego chcemy pobrać wartość indeksujemy od 0!

Return values

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

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

in	*pole*	- numer elementu Listy z którego chcemy pobrać daną

Return values

zwraca	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

in	*dana*	- dana którą chcemy dodać do listy
in	*pole*	- numer elementu listy na który chcemy dodać daną (sieze() 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 znajduję się na Liście

Return values

zwraca	ilosć elementów znadjących 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 oblcizeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypakdu tworzy Listę k elementową wypełnioną stałą liczbą '3'.

Parameters

in	*k*	- ilość danych dla których ma zostać przeprowadzona precedura obn-liczeniowa

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

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania (domyślnie 0 - wysztkie 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]

Wzkaźnik na ostatni element listy.

Wskaźnik na ostatni element listy

4.8.4.2 template<class typ > Element* Lista< typ >::Poczatek [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 Elemenetów znajujących się na Liście

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

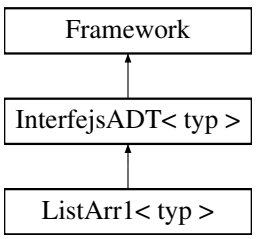
- /home/bartolomeo/209296/prj/inc/Lista.hh

4.9 ListArr1< typ > Class Template Reference

Modeluje pojęcie Listy (array)

`#include <ListArr1.hh>`

Inheritance diagram for ListArr1< typ >:



Public Member Functions

- ListArr1 ()

  Konstruktor bezarumentowy.
- 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

  Wkaźnik 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 bezarumentowy.

Kontruktor 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 którgo chcemy pobrać element

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

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

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	*k*	- 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:

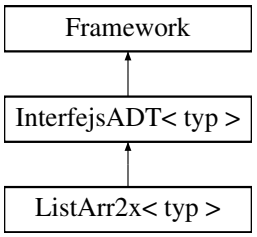
- /home/bartolomeo/209296/prj/inc/ListArr1.hh

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 posortowac tablice przyjmowana jako argument.
- ListArr2x ()
  Konstruktor bezarumentowy.
- 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 wykorzystujaca sortowanei szybkie.
- void QsortOpt (int lewy, int prawy1)
  Zotymalizowane Sortowanie Szybkie.
- void Pokaz ()
  Metoda wypisujaca elemeny 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 wyznaczajaca mediana 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 zamieniajaca Metoda ma za zadanie zamienic miejscami elementy wybrane przez argumenty wywolania.

- int Partycjowanie (int p, int k)

  Metoda segregujaca.

- void Merge (typ *Temp, int l, int s, int p)

  Metoda Dzielaca tablice.

- void Zwolnij ()

  Zwalnia pamięć

- void Kopiec (int Rozmiar, const int i)

  Metoda skladajaca kopiec.

- void BudujKopiec (int Rozmiar)

  Metoda tworzaca kopiec.

Private Attributes

- typ * tab

  Wkaźnik 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 bezarumentowy.

Kontruktor 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 tworzaca kopiec.

Metoda ma za zadanie utworzyc abstrakycjny kopiec z tablicy o podanym poprzez argument rozmiarze

Parameters

in	*Rozmiar*	- Rozmiar kopca

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

Metoda skladajaca kopiec.

Metoda ma za zadanie poprze porownywanie i ustawianie elementow odtworzenie porzadku kopcowego.

Parameters

in	*Rozmiar*	- Rozmar kopca
in	*i*	- indeks ostatniego elementu podzbioru

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

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

Returns

Zwraca indeks na ktorym znajduje sie mediana w tablicy wejsciowej

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

Metoda Dzielaca tablice.

Metoda ma za zadanie przekopiowac zawartosc zbiotu glownego do tablicy tymczasowej.Nastepnie operujac na kopii ustawia wskazniki na poczatki kolejnych zbiorow i porownywane sa wskaane wartosci. Mniejsze wpisujemy do zbioru glownego i przesuwamy odpowiedni wskaznik Czynnos wykonujemy rekurencyjnie az do momentu gdy jeden ze wskaznikow osiagnie koniec zbioru

Parameters

in	*Temp*	- Wskaznik na tablice pomocnicza
in	*l*	- Poczatkowy indeks tablicy
in	*s*	- Srodkowy indeks tablicy
in	*p*	- Koncowy indks tablicy

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

Metoda segregujaca.

Metoda ma za zadanie wybor elementu, ktory ma byc uzyty do podzialu i przenosi wszytskie elementy mniejsze na lewo od tego elementu, a wieksze elementy na prawo od wybranego elementu

Parameters

in	*p*	- poczatkowy indeks podzbiotru
in	*k*	- koncowy indeks podzbioru

Returns

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

Metoda wypisujaca elemeny listy.

Metoda ma za zadanie wypisac wszystkie elementy znajdujace sie obecnie na liscie 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 którgo chcemy pobrać element (indeksowane od 0)

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

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

Implements InterfejsADT< typ >.

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

Metoda wykorzystujaca sortowanei szybkie.

Parameters

in	*l*	- poczatkowy indeks tablicy
in	*h*	- koncowy indeks tablicy

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

Zotymalizowane Sortowanie Szybkie.

Metoda modeluje algorytm sorotwanie szybkiego z zaimplementowanym algorytmem doboru pivotu, tak aby nie zostal wybrany najmniejszy element w danym podzbiorze. [in] lewy - poczatkowy indeks pozbioru

Parameters

in	*prawy*	- koncowy 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 wywolan 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 wywolywane jest sortowanie przez wstawianie, ktore jest wydajne dla tablic o malych rozmiarach

Parameters

in	*l*	- indeks poczatkowego elementu pozbioru
in	*h*	- 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

in	*Rozmiar*	- Rozmiar kopca do zbudowania, ilość danych do posortowania.

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

Metoda testująca czas.

Metoda testująca 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 testująca czas.

Metoda testująca czas wczytania n elementów na ListęArr1

Parameters

in	*k*	- 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 posortowac tablice przyjmowana jako argument.

Parameters

in	T	- Wskaznik na tablice z danymi wejsciowymi
in	l	- Poczatkowy indeks elementu sortowanego podzbioru
in	p	- koncowy indeks elementu sortowanego podzbioru

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

Metoda zamieniajaca Metoda ma za zadanie zamienic miejscami elementy wybrane przez argumenty wywolania.

Parameters

in	i	- Adres elementu podlegajacy zamianie
in	j	- Adres elementu podlegajacy 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]

Wkaźnik 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 pramametrami.
- ~Statystyka ()

  Destruktor - zwaknia pamięć
- double & operator[] (unsigned int i)

  Indeksuje tablicę czasową
- void ZapiszStaty (std::string nazwaPliku)

  Zapisuje statysykę 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 różnyuch wielkości prób.

4.11.2 Constructor & Destructor Documentation

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

Konstruktor z dwoma pramametrami.

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

Parameters

in	*iloscProb*	- liczbosc prob w ksperymencie
in	*proby*	- tablica z licznościami prób.

4.11.2.2 Statystyka::~Statystyka ( ) [inline]

Destruktor - zwaknia pamięć

Zwalnia pamięć zaalokowaną na dynamiczne tablicy 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

in	*i*	- indeks tablicy czasowej

Return values

Czas[i]	referencja do wybranego indeksu

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

Zapisuje statysykę do pliku.

Zapisuje statystystykę 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:

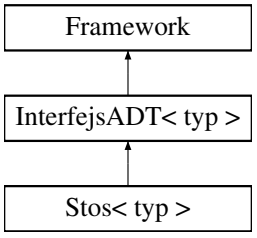
- /home/bartolomeo/209296/prj/inc/Statystyka.hh
- /home/bartolomeo/209296/prj/src/Statystyka.cpp

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 objekt 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

in	*pole*	- 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

in	*dana*	- dana którą chcemy dodać do Stosu
in	*pole*	- 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 znajduję się na Stosie

Return values

zwraca	ilosć elementów znadjących 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 oblcizeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypakdu tworzy Stos k elementowy wypełniony stałą liczbą '3'.

Parameters

in	*k*	- ilość danych dla których ma zostać przeprowadzona precedura obnliczeniowa

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

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

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

Destruktor Stosu.

Zwalnia zaalokowana przez Stos pamiec

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 Elemenetów znajujących 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 przchowywanych 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 przchowywanych 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 ListaArr1. Plik zawiera definicję klasy ListaArr1 ujętej w szablon typu wraz z jej składowymi metofdami.

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 ListaArr2x ujętej w szablon typu wraz z jej składowymi metofdami.

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

Funkcje obslugi plikow.

```
#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 czyszcąc 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 obslugi plikow. Plik zawiera deklaracje funkcji zwiazanych z obsuga plikow

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 podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parameters

in	*n*	- ilość liczb do zapisania
in	*zakres*	- 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 sie powiodlo jezeli nie to koczy program

Parameters

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

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

Otwiera plik do zapisu czyszcąc jego zawartość

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

Parameters

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany 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

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

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

5.10.2 Macro Definition Documentation

5.10.2.1 #define ILOSC_POWTORZEN 10

Ilośc powtórzeń danej próby.

Ilośc 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 obslugi plikow.

```
#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 czyszcąc 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 obslugi plikow. Plik zawiera definicje funkcji zwiazanych z obsluga plikow.

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 podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parameters

in	*n*	- ilość liczb do zapisania
in	*zakres*	- 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	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

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

Otwiera plik do zapisu czyszcąc jego zawartość

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

Parameters

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- 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