

PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Cz, 23 kwi 2015 10:36:49

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	2
2.1 Lista klas	2
3 Indeks plików	2
3.1 Lista plików	2
4 Dokumentacja klas	3
4.1 Dokumentacja szablonu klasy Benchmark< typ >	3
4.1.1 Opis szczegółowy	3
4.1.2 Dokumentacja konstruktora i destruktora	3
4.1.3 Dokumentacja funkcji składowych	4
4.1.4 Dokumentacja atrybutów składowych	4
4.2 Dokumentacja struktury Lista< typ >::Element	4
4.2.1 Opis szczegółowy	5
4.2.2 Dokumentacja konstruktora i destruktora	5
4.2.3 Dokumentacja atrybutów składowych	5
4.3 Dokumentacja klasy Framework	5
4.3.1 Opis szczegółowy	6
4.3.2 Dokumentacja funkcji składowych	6
4.4 Dokumentacja szablonu klasy InterfejsADT< typ >	7
4.4.1 Opis szczegółowy	7
4.4.2 Dokumentacja funkcji składowych	7
4.5 Dokumentacja szablonu klasy Lista< typ >	9
4.5.1 Opis szczegółowy	10
4.5.2 Dokumentacja konstruktora i destruktora	11
4.5.3 Dokumentacja funkcji składowych	11
4.5.4 Dokumentacja atrybutów składowych	12
4.6 Dokumentacja szablonu klasy ListArr2x< typ >	13
4.6.1 Opis szczegółowy	14
4.6.2 Dokumentacja konstruktora i destruktora	14
4.6.3 Dokumentacja funkcji składowych	14
4.6.4 Dokumentacja atrybutów składowych	19
4.7 Dokumentacja klasy Statystyka	20
4.7.1 Opis szczegółowy	20
4.7.2 Dokumentacja konstruktora i destruktora	20
4.7.3 Dokumentacja funkcji składowych	20

4.7.4	Dokumentacja atrybutów składowych	21
5	Dokumentacja plików	21
5.1	Dokumentacja pliku Benchmark.hh	21
5.1.1	Opis szczegółowy	22
5.2	Dokumentacja pliku Framework.hh	22
5.2.1	Opis szczegółowy	22
5.3	Dokumentacja pliku InterfejsADT.hh	22
5.4	Dokumentacja pliku Lista.hh	22
5.4.1	Opis szczegółowy	22
5.5	Dokumentacja pliku ListArr2x.hh	23
5.5.1	Opis szczegółowy	23
5.6	Dokumentacja pliku main.cpp	23
5.6.1	Opis szczegółowy	23
5.6.2	Dokumentacja funkcji	23
5.6.3	Dokumentacja zmiennych	24
5.7	Dokumentacja pliku Pliki.cpp	24
5.7.1	Opis szczegółowy	24
5.7.2	Dokumentacja funkcji	24
5.8	Dokumentacja pliku Pliki.hh	25
5.8.1	Opis szczegółowy	25
5.8.2	Dokumentacja funkcji	25
5.9	Dokumentacja pliku Statystyka.cpp	26
5.9.1	Opis szczegółowy	26
5.10	Dokumentacja pliku Statystyka.hh	26
5.10.1	Opis szczegółowy	26
Indeks		27

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

Benchmark	< typ >	3
Lista	< typ >::Element	4
Framework		5
InterfejsADT	< typ >	7
Lista	< typ >	9
ListArr2x	< typ >	13

Statystyka	20
------------	----

2 Indeks klas

2.1 Lista klas

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

Benchmark < typ > Modeluje pojęcie Benchmarku	3
Lista < typ >::Element Modeluje jeden element Listy	4
Framework Modeluje interfejs programu	5
InterfejsADT < typ >	7
Lista < typ > Modeluje pojęcie listy	9
ListArr2x < typ > Modeluje pojęcie Listy (array)	13
Statystyka Modeluje pojęcie statystyki	20

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Benchmark.hh Definicja klasy Benchmark	21
Framework.hh Definicja klasy Framework	22
InterfejsADT.hh	22
Lista.hh Definicja klasy Lista	22
ListArr2x.hh Definicja klasy ListArr2x	23
main.cpp Moduł główny programu	23
Pliki.cpp Definicje funkcji obsługi plików	24
Pliki.hh Funkcje obsługi plików	25

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

26

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

26

4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy `Benchmark< typ >`

Modeluje pojęcie Benchmarku.

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

Metody publiczne

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

Atrybuty prywatne

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

4.1.1 Opis szczegółowy

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

Modeluje pojęcie Benchmarku czyli obiektu mierzącego czas wykonywania algoytmu

Definicja w linii 24 pliku Benchmark.hh.

4.1.2 Dokumentacja konstruktora i destruktora

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

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

Parametry

<code>in</code>	<code>ileProb</code>	- ilość prób, które zostaną wykonane
-----------------	----------------------	--------------------------------------

in	<i>ileDanych</i>	- wskaźnik na tablice z licznosciami kolejnych serii
in	<i>ilePowtorzen</i>	- ilość powtórzeń każdej serii

Definicja w linii 69 pliku Benchmark.hh.

4.1.3 Dokumentacja funkcji składowych

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

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

Parametry

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

Definicja w linii 86 pliku Benchmark.hh.

4.1.4 Dokumentacja atrybutów składowych

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

Tablica z licznosciami elementów dla kolejnych serii

Definicja w linii 47 pliku Benchmark.hh.

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

Ilość powtórzeń każdej serii

Definicja w linii 55 pliku Benchmark.hh.

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

Ilość powtórzeń każdej serii

Definicja w linii 39 pliku Benchmark.hh.

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

Pole przechowuje wyniki testów

Definicja w linii 31 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Benchmark.hh](#)

4.2 Dokumentacja struktury `Lista< typ >::Element`

Modeluje jeden element Listy.

Metody publiczne

- [Element](#) (typ k)

Konstruktor daną przekazywaną w argumencie.

Atrybuty publiczne

- typ [wartosc](#)

Wartosc Elementu.

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

Wskaźnik na kolejny [Element](#) Listy.

4.2.1 Opis szczegółowy

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

Modeluje jeden nierozłączny element listy - przechowywaną daną oraz wskaźnik na następny element;
Definicja w linii 33 pliku Lista.hh.

4.2.2 Dokumentacja konstruktora i destruktor

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

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

Parametry

in	k	- dana która ma zostać dodana na koniec Listy
----	---	---

Definicja w linii 59 pliku Lista.hh.

4.2.3 Dokumentacja atrybutów składowych

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

Wskaźnik na kolejny [Element](#) Listy

Definicja w linii 48 pliku Lista.hh.

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

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

Definicja w linii 41 pliku Lista.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

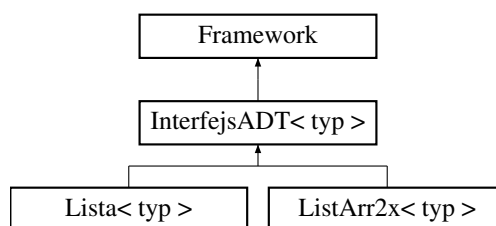
- [Lista.hh](#)

4.3 Dokumentacja klasy Framework

Modeluje interfejs programu.

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

Diagram dziedziczenia dla Framework



Metody publiczne

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

4.3.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanych w ramach kursu.

Definicja w linii 24 pliku Framework.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 virtual void Framework::Pokaz () [pure virtual]

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

4.3.2.2 virtual void Framework::Start (const unsigned int k) [pure virtual]

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

Parametry

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

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

4.3.2.3 virtual void Framework::WczytajDane (const char * nazwaPliku, unsigned int n) [pure virtual]

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

Parametry

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

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

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

Zwalnia pamięć zajmowaną przez obiekty wykorzystane do testów

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

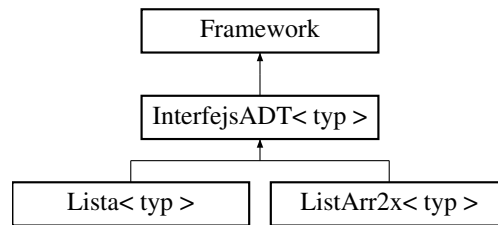
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Framework.hh](#)

4.4 Dokumentacja szablonu klasy InterfejsADT< typ >

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

Diagram dziedziczenia dla InterfejsADT< typ >



Metody publiczne

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

4.4.1 Opis szczegółowy

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

\ brief Definiuje interfejs użytkownika

Definiuje interfejs użytkownika dla listy, stosu i kolejki.

Definicja w linii 13 pliku InterfejsADT.hh.

4.4.2 Dokumentacja funkcji składowych

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

Pobiera element z typu danych

Parametry

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

Zwracane wartości

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

Implementowany w `ListArr2x< typ >` i `Lista< typ >`.

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

Dodaje kolejny element do typu danych

Parametry

in	<i>dana</i>	- element który chcemy dorzucić do naszego typu
in	<i>pole</i>	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola na które chcemy dodać element

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

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

Informuje o liczności elementów obecnie przechowywanych

Zwracane wartości

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

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

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

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

Parametry

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

Implementuje [Framework](#).

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

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

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

Parametry

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

Implementuje [Framework](#).

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

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

Zwalnia pamięć zajmowaną przez daną strukturę

Implementuje [Framework](#).

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

Dokumentacja dla tej klasy została wygenerowana z pliku:

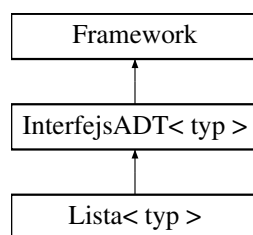
- [InterfejsADT.hh](#)

4.5 Dokumentacja szablonu klasy Lista< typ >

Modeluje pojęcie listy.

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

Diagram dziedziczenia dla Lista< typ >



Komponenty

- struct `Element`
Modeluje jeden element Listy.

Metody publiczne

- `Lista ()`
Konstruktor puste listy.
- void `Zwolnij ()`
Destruktor listy.
- void `push (const typ dana, const unsigned int pole)`
Dodaje daną do Listy.
- typ `pop (const unsigned int pole)`
Usuwa element z Listy.
- unsigned int `size () const`
Sprawdza rozmiar Listy.
- void `WczytajDane (const char *nazwaPliku, unsigned int n=0)`
Wczytuje dane z pliku.
- typ `operator[] (const size_t pole) const`
Wyciąga wartość elementu Listy.
- void `Start (const unsigned int k)`
Proces obliczeniowy.

Atrybuty prywatne

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

4.5.1 Opis szczegółowy

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

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

Definicja w linii 24 pliku Lista.hh.

4.5.2 Dokumentacja konstruktora i destruktora

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

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

Definicja w linii 98 pliku Lista.hh.

4.5.3 Dokumentacja funkcji składowych

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

Wyluskuje wartość danego elementu z Listy

Parametry

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

Zwracane wartości

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

Definicja w linii 284 pliku Lista.hh.

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

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

Parametry

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

Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 190 pliku Lista.hh.

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

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

Parametry

<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 (sieze() jeżeli na koniec)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 142 pliku Lista.hh.

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

Sprawdza ile aktualnie elementów znajdują się na Liście

Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 240 pliku Lista.hh.

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 306 pliku Lista.hh.

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

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

Parametry

<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)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 254 pliku Lista.hh.

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

Zwalnia zaalokowana przez listę pamięć

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez listę

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 122 pliku Lista.hh.

4.5.4 Dokumentacja atrybutów składowych

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

Wskaźnik na ostatni element listy

Definicja w linii 79 pliku Lista.hh.

4.5.4.2 `template<class typ> Element* Lista< typ >::Poczatek [private]`

Wskaźnik na pierwszy element Listy

Definicja w linii 71 pliku Lista.hh.

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

Przechowuje aktualną ilość Elementów znajdujących się na Liście

Definicja w linii 86 pliku Lista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

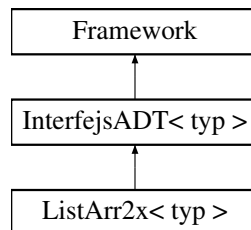
- [Lista.hh](#)

4.6 Dokumentacja szablonu klasy ListArr2x< typ >

Modeluje pojęcie Listy (array)

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

Diagram dziedziczenia dla ListArr2x< typ >



Metody publiczne

- [ListArr2x](#) ()
Konstruktor bezargumentowy.
- void [push](#) (const typ dana, const unsigned int pole)
Dodaje element do ListyArr2x.
- typ [pop](#) (const unsigned int pole)
Pobiera element z ListyArr2x.
- unsigned int [size](#) () const
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
- void [HeapSort](#) (int rozmiar)
Sortowanie przez kopcowanie.
- void [Pokaz](#) ()
Wyświetla elementy Listy.
- void [QSortOpt](#) (const int lewy, const int prawy)
Zoptymalizowane Szybkie Sortowanie.
- void [InsertSort](#) (int pierwszyElement, int ostatniElement)
Sortowanie przez wstawianie.
- void [HybridSort](#) (int lewy, int prawy)
Sortowanie hybrydowe.

Metody prywatne

- void **Zamien** (typ &a, typ &b)
Zamienia elementy publicz.
- void **Kopcuje** (const int rozmiarKopca, const int i)
Porównuje el. kopca.
- void **BudujKopiec** (const int rozmiar)
Tworzy kopiec.
- int **MedianaTrzech** (const int a, const int b, const int c) const
Znajduje mediane.
- int **Partition** (int lewy, int prawy)

Atrybuty prywatne

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

4.6.1 Opis szczegółowy

```
template<class typ>class ListArr2x< typ >
```

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

Definicja w linii 21 pliku ListArr2x.hh.

4.6.2 Dokumentacja konstruktora i destruktor

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

Konstruktor alokujący tablicę jednoelementową z której będzie tworzona lista

Definicja w linii 162 pliku ListArr2x.hh.

4.6.3 Dokumentacja funkcji składowych

```
4.6.3.1 template<class typ> void ListArr2x< typ >::BudujKopiec ( const int rozmiar ) [inline], [private]
```

Tworzy kopiec z tablicy o podanym rozmiarze

Parametry

in	rozmiar	- rozmiar tablicy
----	---------	-------------------

Definicja w linii 96 pliku ListArr2x.hh.

```
4.6.3.2 template<class typ> void ListArr2x< typ >::HeapSort ( int rozmiar ) [inline]
```

Realizuje algorytm sortowania przez kopcowanie

Parametry

in	<i>rozmiar</i>	- rozmiar tablicy do posortowania
----	----------------	-----------------------------------

Definicja w linii 337 pliku ListArr2x.hh.

4.6.3.3 `template<class typ> void ListArr2x< typ >::HybridSort (int lewy, int prawy) [inline]`

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

Parametry

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

Definicja w linii 409 pliku ListArr2x.hh.

4.6.3.4 `template<class typ> void ListArr2x< typ >::InsertSort (int pierwszyElement, int ostatniElement) [inline]`

Metoda realizuje algorytm sortowania przez wstawianie.

Parametry

in	<i>pierwszyElement</i>	- indeks pierwszego elementu do posortowania
in	<i>ostatniElement</i>	- indeks ostatniego elementu do posortowania

Definicja w linii 386 pliku ListArr2x.hh.

4.6.3.5 `template<class typ> void ListArr2x< typ >::Kopuj (const int rozmiarKopca, const int i) [inline], [private]`

Porównuje i ustawia elementy kopca w odpowiedniej kolejności

Parametry

in	<i>rozmiarKopca</i>	- rozmiar kopca który sortujemy
in	<i>i</i>	- numer gałęzi kopca

Definicja w linii 70 pliku ListArr2x.hh.

4.6.3.6 `template<class typ> int ListArr2x< typ >::MedianaTrzech (const int a, const int b, const int c) const [inline], [private]`

Znajduje mediane wartości z trzech podanych elementów Listy

Parametry

in	<i>a</i>	- indeks pierwszego elementu do liczenia mediany
in	<i>b</i>	- indeks drugiego elementu do liczenia mediany
in	<i>c</i>	- indeks trzeciego elementu do liczenia mediany

Zwracane wartości

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

Definicja w linii 113 pliku ListArr2x.hh.

4.6.3.7 `template<class typ> int ListArr2x< typ >::Partition (int lewy, int prawy) [inline], [private]`

Partycjonowanie listy

Metoda będąca częścią algorytmu Sortowania Szybkiego. Dzieli przekazany fragment listy na dwie części - lewy z elementami mniejszymi od wybranego pivota i prawa z elementami większymi od wybranego pivota. Pivot jest

dobierany za pomocą liczenia mediany z trzech elementów: pierwszego, środkowego i ostatniego.

Parametry

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

Definicja w linii 139 pliku ListArr2x.hh.

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

Metoda wypsuje na terminal elementy znajdujące się na liście

Implementuje [Framework](#).

Definicja w linii 351 pliku ListArr2x.hh.

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

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

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

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 229 pliku ListArr2x.hh.

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

Dodaje nowy element do ListyArr2x

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 179 pliku ListArr2x.hh.

4.6.3.11 `template<class typ> void ListArr2x< typ >::QSortOpt (const int lewy, const int prawy) [inline]`

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

Parametry

in	<i>lewy</i>	- indeks pierwszego elementu tworzącego Listę do posortowania
in	<i>prawy</i>	- indeks ostatniego elementu tworzącego Listę do posortowania

Definicja w linii 366 pliku ListArr2x.hh.

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

Informuje o ilości elementów znajdujących się na LiścieArr2x

Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 279 pliku ListArr2x.hh.

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

Metoda testująca czas wczytania n elementów na ListęArr2x

Parametry

in	k	- ilość elementów do wczytania
----	---	--------------------------------

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 289 pliku ListArr2x.hh.

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

Wczytuje dane z pliku do [ListArr2x](#)

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 305 pliku ListArr2x.hh.

4.6.3.15 `template<class typ> void ListArr2x< typ >::Zamien (typ & a, typ & b)` `[inline], [private]`

Zamienia dwa elementy tablicy o polach podanych w wywołaniu

Parametry

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

Definicja w linii 55 pliku ListArr2x.hh.

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

Zwalnia pamięć zaalokowaną przez [ListArr2x](#)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 322 pliku ListArr2x.hh.

4.6.4 Dokumentacja atrybutów składowych

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

Aktualny rozmiar ListyArr2x

Definicja w linii 45 pliku ListArr2x.hh.

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

Aktualny rozmiar tablicy.

Definicja w linii 37 pliku ListArr2x.hh.

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

Wskaźnik na dynamiczną tablicę tworzącą ListęArr2x

Definicja w linii 29 pliku ListArr2x.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [ListArr2x.hh](#)

4.7 Dokumentacja klasy Statystyka

Modeluje pojęcie statystyki.

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

Metody publiczne

- **Statystyka** (const unsigned int iloscProb, unsigned int *proby)
Konstruktor z dwoma parametrami.
- **~Statystyka** ()
Destruktor - zwalnia pamięć
- double & **operator[]** (unsigned int i)
Indeksuje tablicę czasową
- void **ZapiszStaty** (std::string nazwaPliku)
Zapisuje statystykę do pliku.

Atrybuty prywatne

- unsigned int **ileProb**
Ilość prób.
- unsigned int * **Proba**
Tablica z rozmiarami prób.
- double * **Czas**
Średni czas wykonania danej próby.

4.7.1 Opis szczegółowy

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

Definicja w linii 22 pliku Statystyka.hh.

4.7.2 Dokumentacja konstruktora i destruktora

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

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

Parametry

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

Definicja w linii 14 pliku Statystyka.cpp.

4.7.2.2 Statystyka::~Statystyka () [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablice przechowujące statystykę.

Definicja w linii 68 pliku Statystyka.hh.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 double& Statystyka::operator[] (unsigned int *i*) [inline]

Zwraca referencję do i-tego indeksu tablicy czasowej.

Parametry

in	i	- indeks tablicy czasowej
----	---	---------------------------

Zwracane wartości

Czas[i]	referencja do wybranego indeksu
---------	---------------------------------

Definicja w linii 80 pliku Statystyka.hh.

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

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

Definicja w linii 22 pliku Statystyka.cpp.

4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 46 pliku Statystyka.hh.

4.7.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 30 pliku Statystyka.hh.

4.7.4.3 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 38 pliku Statystyka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

5 Dokumentacja plików

5.1 Dokumentacja pliku Benchmark.hh

Definicja klasy [Benchmark](#).

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

Komponenty

- class [Benchmark](#)< typ >

Modeluje pojęcie Benchmarku.

5.1.1 Opis szczegółowy

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

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

5.2 Dokumentacja pliku Framework.hh

Definicja klasy [Framework](#).

```
#include <iostream>
```

Komponenty

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

5.2.1 Opis szczegółowy

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

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

5.3 Dokumentacja pliku InterfejsADT.hh

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

Komponenty

- class [InterfejsADT< typ >](#)

5.4 Dokumentacja pliku Lista.hh

Definicja klasy [Lista](#).

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

Komponenty

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

5.4.1 Opis szczegółowy

Plik zawiera definicję klasy lista ujętej w szablon typu przechowywanych zmiennych więc zawiera też definicję metod klasy.

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

5.5 Dokumentacja pliku ListArr2x.hh

Definicja klasy [ListArr2x](#).

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

Komponenty

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

5.5.1 Opis szczegółowy

Plik zawiera definicję klasy [ListaArr2x](#) ujętej w szablon typu wraz z jej składowymi metodami.

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

5.6 Dokumentacja pliku main.cpp

Moduł główny programu.

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

Funkcje

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

Zmienne

- const int [ILOSC_POWTORZEN](#) = 100
Ilość powtórzeń danej próby.
- const int [ILOSC_PROB](#) = 11
Ilość prób.

5.6.1 Opis szczegółowy

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 "Typ-Daych.dat". gdzie "TypDanych" to odpowiednio [Lista](#), ListaArr1 i ListaArr2x

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

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

5.6.2 Dokumentacja funkcji

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

Definicja w linii 42 pliku main.cpp.

5.6.3 Dokumentacja zmiennych

5.6.3.1 `const int ILOSC_POWTORZEN = 100`

Ilość powtórzeń danej próby

Definicja w linii 32 pliku `main.cpp`.

5.6.3.2 `const int ILOSC_PROB = 11`

Ilość prób = ilość rozmiarów prób

Definicja w linii 40 pliku `main.cpp`.

5.7 Dokumentacja pliku `Pliki.cpp`

Definicje funkcji obsługi plików.

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

Funkcje

- void `OtworzPlikIn` (const char *nazwaPliku, std::fstream &plik)
Otwiera plik do odczytu.
- void `OtworzPlikOut` (const char *nazwaPliku, std::fstream &plik)
Otwiera plik do zapisu czyszcząc jego zawartość
- void `LosujIntDoPliku` (const unsigned int n, const unsigned int zakres)
Zapisuje n losowych liczb(int) do pliku.

5.7.1 Opis szczegółowy

Plik zawiera definicje funkcji związanych z obsługą plików.

Definicja w pliku `Pliki.cpp`.

5.7.2 Dokumentacja funkcji

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

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

Parametry

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

Definicja w linii 27 pliku `Pliki.cpp`.

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

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

Parametry

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

Definicja w linii 11 pliku Pliki.cpp.

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

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

Parametry

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

Definicja w linii 19 pliku Pliki.cpp.

5.8 Dokumentacja pliku Pliki.hh

Funkcje obsługi plików.

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

Funkcje

- void [OtworzPlikIn](#) (const char **nazwaPliku*, std::fstream &*plik*)
Otwiera plik do odczytu.
- void [OtworzPlikOut](#) (const char **nazwaPliku*, std::fstream &*plik*)
Otwiera plik do zapisu czyszcząc jego zawartość
- void [LosujIntDoPliku](#) (const unsigned int *n*, const unsigned int *zakres*)
*Zapisuje *n* losowych liczb(int) do pliku.*

5.8.1 Opis szczegółowy

Plik zawiera deklaracje funkcji związanych z obsługą plików

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

5.8.2 Dokumentacja funkcji

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

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

Parametry

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

Definicja w linii 27 pliku Pliki.cpp.

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

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

Parametry

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

Definicja w linii 11 pliku *Pliki.cpp*.

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

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

Parametry

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

Definicja w linii 19 pliku *Pliki.cpp*.

5.9 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy [Statystyka](#).

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

5.9.1 Opis szczegółowy

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

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

5.10 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy [Statystyka](#).

```
#include <iostream>
```

Komponenty

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

5.10.1 Opis szczegółowy

Zawiera definicję klasy [Statystyka](#)

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

Skorowidz

~Statystyka

Statystyka, 20

Benchmark

Benchmark, 3

IleDanych, 4

IlePowtorzen, 4

IleProb, 4

stat, 4

Test, 4

Benchmark< typ >, 3

Benchmark.hh, 21

BudujKopiec

ListArr2x, 14

Czas

Statystyka, 21

Element

Lista::Element, 5

Framework, 5

Pokaz, 6

Start, 6

WczytajDane, 6

Zwolnij, 6

Framework.hh, 22

HeapSort

ListArr2x, 14

HybridSort

ListArr2x, 15

ILOSC_POWTORZEN

main.cpp, 24

ILOSC_PROB

main.cpp, 24

IleDanych

Benchmark, 4

IlePowtorzen

Benchmark, 4

IleProb

Benchmark, 4

Statystyka, 21

InsertSort

ListArr2x, 15

InterfejsADT

pop, 7

push, 7

size, 9

Start, 9

WczytajDane, 9

Zwolnij, 9

InterfejsADT< typ >, 7

InterfejsADT.hh, 22

Koniec

Lista, 12

Kopcuje

ListArr2x, 15

ListArr2x

BudujKopiec, 14

HeapSort, 14

HybridSort, 15

InsertSort, 15

Kopcuje, 15

ListArr2x, 14

ListArr2x, 14

MedianaTrzech, 15

Partition, 15

Pokaz, 17

pop, 17

push, 17

QSortOpt, 17

RozmiarL, 19

RozmiarT, 19

size, 17

Start, 17

tab, 19

WczytajDane, 19

Zamien, 19

Zwolnij, 19

ListArr2x< typ >, 13

ListArr2x.hh, 23

Lista

Koniec, 12

Lista, 11

Poczatek, 12

pop, 11

push, 11

Rozmiar, 12

size, 11

Start, 12

WczytajDane, 12

Zwolnij, 12

Lista< typ >, 9

Lista< typ >::Element, 4

Lista.hh, 22

Lista::Element

Element, 5

nastepny, 5

wartosc, 5

LosujIntDoPliku

Pliki.cpp, 24

Pliki.hh, 25

main

main.cpp, 23

main.cpp, 23

ILOSC_POWTORZEN, 24

ILOSC_PROB, 24

main, 23

MedianaTrzech
 ListArr2x, 15

nastepny
 Lista::Element, 5

OtworzPlikIn
 Pliki.cpp, 24
 Pliki.hh, 25

OtworzPlikOut
 Pliki.cpp, 25
 Pliki.hh, 26

Partition
 ListArr2x, 15

Pliki.cpp, 24
 LosujIntDoPliku, 24
 OtworzPlikIn, 24
 OtworzPlikOut, 25

Pliki.hh, 25
 LosujIntDoPliku, 25
 OtworzPlikIn, 25
 OtworzPlikOut, 26

Poczatek
 Lista, 12

Pokaz
 Framework, 6
 ListArr2x, 17

pop
 InterfejsADT, 7
 Lista, 11
 ListArr2x, 17

Proba
 Statystyka, 21

push
 InterfejsADT, 7
 Lista, 11
 ListArr2x, 17

QSortOpt
 ListArr2x, 17

Rozmiar
 Lista, 12

RozmiarL
 ListArr2x, 19

RozmiarT
 ListArr2x, 19

size
 InterfejsADT, 9
 Lista, 11
 ListArr2x, 17

Start
 Framework, 6
 InterfejsADT, 9
 Lista, 12
 ListArr2x, 17

stat
 Benchmark, 4

Statystyka, 20
 ~Statystyka, 20
 Czas, 21
 IleProb, 21
 Proba, 21
 Statystyka, 20
 ZapiszStaty, 21

Statystyka.cpp, 26
Statystyka.hh, 26

tab
 ListArr2x, 19

Test
 Benchmark, 4

wartosc
 Lista::Element, 5

WczytajDane
 Framework, 6
 InterfejsADT, 9
 Lista, 12
 ListArr2x, 19

Zamien
 ListArr2x, 19

ZapiszStaty
 Statystyka, 21

Zwolnij
 Framework, 6
 InterfejsADT, 9
 Lista, 12
 ListArr2x, 19