

PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Cz, 19 mar 2015 09:53:23

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	2
2 Indeks klas	2
2.1 Lista klas	2
3 Indeks plików	2
3.1 Lista plików	3
4 Dokumentacja klas	3
4.1 Dokumentacja struktury Kolejka< typ >::Element	3
4.1.1 Opis szczegółowy	3
4.1.2 Dokumentacja konstruktora i destruktora	4
4.1.3 Dokumentacja atrybutów składowych	4
4.2 Dokumentacja struktury Stos< typ >::Element	4
4.2.1 Opis szczegółowy	4
4.2.2 Dokumentacja konstruktora i destruktora	5
4.2.3 Dokumentacja atrybutów składowych	5
4.3 Dokumentacja struktury Lista< typ >::Element	5
4.3.1 Opis szczegółowy	5
4.3.2 Dokumentacja konstruktora i destruktora	6
4.3.3 Dokumentacja atrybutów składowych	6
4.4 Dokumentacja klasy Framework	6
4.4.1 Opis szczegółowy	6
4.4.2 Dokumentacja funkcji składowych	7
4.5 Dokumentacja szablonu klasy InterfejsADT< typ >	7
4.5.1 Opis szczegółowy	8
4.5.2 Dokumentacja funkcji składowych	8
4.6 Dokumentacja szablonu klasy Kolejka< typ >	9
4.6.1 Opis szczegółowy	10
4.6.2 Dokumentacja konstruktora i destruktora	10
4.6.3 Dokumentacja funkcji składowych	10
4.6.4 Dokumentacja atrybutów składowych	11
4.7 Dokumentacja szablonu klasy Lista< typ >	12
4.7.1 Opis szczegółowy	13
4.7.2 Dokumentacja konstruktora i destruktora	13
4.7.3 Dokumentacja funkcji składowych	13
4.7.4 Dokumentacja atrybutów składowych	14
4.8 Dokumentacja klasy Statystyka	14

4.8.1	Opis szczegółowy	15
4.8.2	Dokumentacja konstruktora i destruktora	15
4.8.3	Dokumentacja funkcji składowych	15
4.8.4	Dokumentacja atrybutów składowych	16
4.9	Dokumentacja szablonu klasy Stos< typ >	16
4.9.1	Opis szczegółowy	17
4.9.2	Dokumentacja konstruktora i destruktora	17
4.9.3	Dokumentacja funkcji składowych	17
4.9.4	Dokumentacja atrybutów składowych	18
5	Dokumentacja plików	19
5.1	Dokumentacja pliku Framework.hh	19
5.1.1	Opis szczegółowy	19
5.2	Dokumentacja pliku InterfejsADT.hh	19
5.3	Dokumentacja pliku Kolejka.cpp	19
5.3.1	Opis szczegółowy	20
5.3.2	Dokumentacja definicji	20
5.4	Dokumentacja pliku Lista.cpp	20
5.4.1	Opis szczegółowy	20
5.4.2	Dokumentacja definicji	20
5.5	Dokumentacja pliku main.cpp	21
5.5.1	Opis szczegółowy	21
5.5.2	Dokumentacja definicji	21
5.5.3	Dokumentacja funkcji	21
5.6	Dokumentacja pliku Pliki.cpp	21
5.6.1	Opis szczegółowy	22
5.6.2	Dokumentacja funkcji	22
5.7	Dokumentacja pliku Pliki.hh	22
5.7.1	Opis szczegółowy	23
5.7.2	Dokumentacja funkcji	23
5.8	Dokumentacja pliku Statystyka.cpp	23
5.8.1	Opis szczegółowy	23
5.9	Dokumentacja pliku Statystyka.hh	23
5.9.1	Opis szczegółowy	24
5.10	Dokumentacja pliku Stos.cpp	24
5.10.1	Opis szczegółowy	24
Indeks		25

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

Kolejka< typ >::Element	3
Stos< typ >::Element	4
Lista< typ >::Element	5
Framework	6
InterfejsADT< typ >	7
Kolejka< typ >	9
Lista< typ >	12
Stos< typ >	16
Statystyka	14

2 Indeks klas

2.1 Lista klas

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

Kolejka< typ >::Element Modeluje jeden element Kolejki	3
Stos< typ >::Element Modeluje jeden element Stosu	4
Lista< typ >::Element Modeluje jeden element Listy	5
Framework Modeluje interfejs programu	6
InterfejsADT< typ >	7
Kolejka< typ > Modeluje pojęcie Kolejki	9
Lista< typ > Modeluje pojęcie listy	12
Statystyka Modeluje pojęcie statystyki	14
Stos< typ > Modeluje pojęcie Stosu	16

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Framework.hh	
Definicja klasy Framework	19
InterfejsADT.hh	19
Kolejka.cpp	
Definicja klasy Kolejka	19
Lista.cpp	
Definicja klasy Lista	20
main.cpp	
Moduł główny programu	21
Pliki.cpp	
Definicje funkcji obsługi plików	21
Pliki.hh	
Funkcje obsługi plików	22
Statystyka.cpp	
Zawiera definicję metod klasy Statystyka	23
Statystyka.hh	
Zawiera definicję klasy Statystyka	23
Stos.cpp	
Zawiera definicję Stosu	24

4 Dokumentacja klas

4.1 Dokumentacja struktury `Kolejka< typ >::Element`

Modeluje jeden element Kolejki.

Metody publiczne

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

Atrybuty publiczne

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

4.1.1 Opis szczegółowy

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

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

Definicja w linii 34 pliku Kolejka.cpp.

4.1.2 Dokumentacja konstruktora i destruktora

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

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

Parametry

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

Definicja w linii 60 pliku Kolejka.cpp.

4.1.3 Dokumentacja atrybutów składowych

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

Wskaźnik na kolejny [Element](#) Kolejki

Definicja w linii 49 pliku Kolejka.cpp.

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

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

Definicja w linii 42 pliku Kolejka.cpp.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Kolejka.cpp](#)

4.2 Dokumentacja struktury Stos< typ >::Element

Modeluje jeden element Stosu.

Metody publiczne

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

Atrybuty publiczne

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

4.2.1 Opis szczegółowy

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

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

Definicja w linii 27 pliku Stos.cpp.

4.2.2 Dokumentacja konstruktora i destruktor

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

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

Parametry

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

Definicja w linii 53 pliku Stos.cpp.

4.2.3 Dokumentacja atrybutów składowych

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

Wskaźnik na kolejny [Element](#) Stosu

Definicja w linii 42 pliku Stos.cpp.

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

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

Definicja w linii 35 pliku Stos.cpp.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Stos.cpp](#)

4.3 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.

4.3.1 Opis szczegółowy

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

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

Definicja w linii 34 pliku Lista.cpp.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 `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

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

Definicja w linii 60 pliku Lista.cpp.

4.3.3 Dokumentacja atrybutów składowych

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

Wskaźnik na kolejny [Element](#) Listy

Definicja w linii 49 pliku Lista.cpp.

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

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

Definicja w linii 42 pliku Lista.cpp.

Dokumentacja dla tej struktury została wygenerowana z pliku:

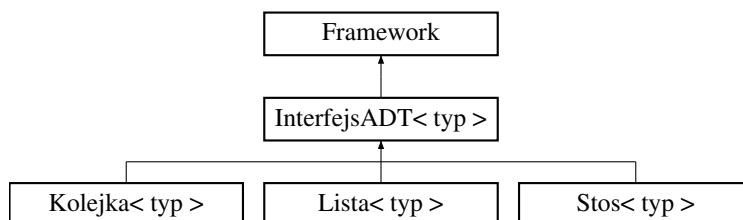
- [Lista.cpp](#)

4.4 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.

4.4.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanych w ramach kursu.

Definicja w linii 24 pliku Framework.hh.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 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 >](#), [Kolejka< typ >](#), [Stos< typ >](#) i [InterfejsADT< typ >](#).

4.4.2.2 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 [Lista< typ >](#), [Kolejka< typ >](#), [Stos< typ >](#) i [InterfejsADT< typ >](#).

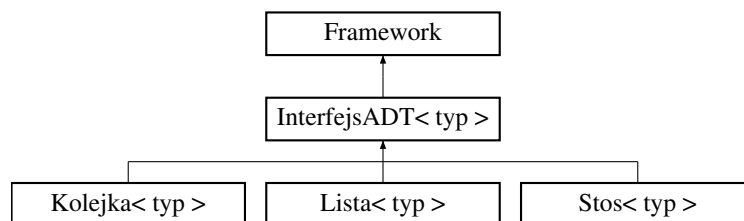
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Framework.hh](#)

4.5 Dokumentacja szablonu klasy InterfejsADT< typ >

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

Diagram dziedziczenia dla InterfejsADT< typ >



Metody publiczne

- virtual void [push](#) (typ dana, unsigned int pole)=0
Dodaje kolejny element.
- virtual void [pop](#) (unsigned int pole)=0
Pobiera element.
- virtual unsigned int [size](#) ()=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

4.5.1 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.

4.5.2 Dokumentacja funkcji składowych

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

Pobiera element z typu danych

Parametry

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

Zwracane wartości

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

Implementowany w [Lista< typ >](#), [Kolejka< typ >](#) i [Stos< typ >](#).

4.5.2.2 `template<class typ> virtual void InterfejsADT< typ >::push (typ dana, 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>	- !!!DOSTĘPNE TYLKO DLA LISTY!!! nr pola na które chcemy dodać element

Implementowany w [Kolejka< typ >](#), [Lista< typ >](#) i [Stos< typ >](#).

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

Informuje o liczności elementów obecnie przechowywanych

Zwracane wartości

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

Implementowany w [Lista< typ >](#), [Kolejka< typ >](#) i [Stos< typ >](#).

4.5.2.4 `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 >](#), [Kolejka< typ >](#) i [Stos< typ >](#).

4.5.2.5 `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 [Lista< typ >](#), [Kolejka< typ >](#) i [Stos< typ >](#).

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

Zwalnia pamięć zajmowaną przez daną strukturę

Implementowany w [Kolejka< typ >](#), [Lista< typ >](#) i [Stos< typ >](#).

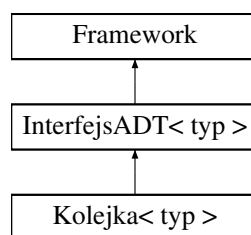
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [InterfejsADT.hh](#)

4.6 Dokumentacja szablonu klasy Kolejka< typ >

Modeluje pojęcie Kolejki.

Diagram dziedziczenia dla Kolejka< typ >



Komponenty

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

Metody publiczne

- [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.

Atrybuty prywatne

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

4.6.1 Opis szczegółowy

```
template<class typ>class Kolejka< typ >
```

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

Definicja w linii 25 pliku Kolejka.cpp.

4.6.2 Dokumentacja konstruktora i destruktor

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

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

Definicja w linii 100 pliku Kolejka.cpp.

4.6.3 Dokumentacja funkcji składowych

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 173 pliku Kolejka.cpp.

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 146 pliku Kolejka.cpp.

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

Sprawdza ile aktualnie elementów znajduje się w Kolejce

Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 194 pliku Kolejka.cpp.

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 220 pliku Kolejka.cpp.

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 206 pliku Kolejka.cpp.

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

Zwalnia zaalokowaną przez Kolejke pamięć

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez Kolejke

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 124 pliku Kolejka.cpp.

4.6.4 Dokumentacja atrybutów składowych

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

Wskaźnik na ostatni element kolejki zwiększający szybkość dodawania danych na końcu

Definicja w linii 81 pliku Kolejka.cpp.

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

Wskaźnik na pierwszy element Kolejki

Definicja w linii 72 pliku Kolejka.cpp.

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

Przechowuje aktualną ilość Elementów znajdujących się w Kolejce

Definicja w linii 88 pliku Kolejka.cpp.

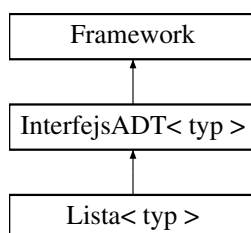
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Kolejka.cpp](#)

4.7 Dokumentacja szablonu klasy Lista< typ >

Modeluje pojęcie listy.

Diagram dziedziczenia dla Lista< typ >



Komponenty

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

Metody publiczne

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

Atrybuty prywatne

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

4.7.1 Opis szczegółowy

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

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

Definicja w linii 25 pliku Lista.cpp.

4.7.2 Dokumentacja konstruktora i destruktor

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

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

Definicja w linii 99 pliku Lista.cpp.

4.7.3 Dokumentacja funkcji składowych

```
4.7.3.1 template<class typ> void Lista< typ >::pop ( 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

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

Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 191 pliku Lista.cpp.

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

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

Parametry

in	<i>dana</i>	- dana którą chcemy dodać do listy
in	<i>pole</i>	- numer elementu listy na który chcemy dodać daną

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 143 pliku Lista.cpp.

```
4.7.3.3 template<class typ> unsigned int Lista< typ >::size ( ) [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 232 pliku Lista.cpp.

```
4.7.3.4 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 wypadku tworzy Listę k elementową wypełnioną stałą liczbą '3'.

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 258 pliku Lista.cpp.

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 244 pliku Lista.cpp.

4.7.3.6 `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 123 pliku Lista.cpp.

4.7.4 Dokumentacja atrybutów składowych

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

Wskaźnik na ostatni element listy

Definicja w linii 80 pliku Lista.cpp.

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

Wskaźnik na pierwszy element Listy

Definicja w linii 72 pliku Lista.cpp.

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

Przechowuje aktualną ilość Elementów znajdujących się na Liście

Definicja w linii 87 pliku Lista.cpp.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Lista.cpp](#)

4.8 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.

4.8.1 Opis szczegółowy

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

Definicja w linii 22 pliku Statystyka.hh.

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 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.

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

Zwalnia pamięć zaalokowaną na dynamiczne tablice przechowujące statystykę.

Definicja w linii 68 pliku Statystyka.hh.

4.8.3 Dokumentacja funkcji składowych

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

Zwraca referencję do i-tego indeksu tablicy czasowej.

Parametry

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

Zwracane wartości

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

Definicja w linii 80 pliku Statystyka.hh.

4.8.3.2 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.

4.8.4 Dokumentacja atrybutów składowych

4.8.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 46 pliku Statystyka.hh.

4.8.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 30 pliku Statystyka.hh.

4.8.4.3 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 38 pliku Statystyka.hh.

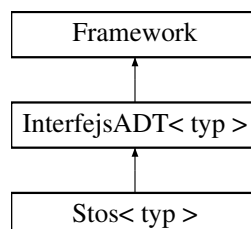
Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.9 Dokumentacja szablonu klasy Stos< typ >

Modeluje pojęcie Stosu.

Diagram dziedziczenia dla Stos< typ >

**Komponenty**

- struct [Element](#)

Modeluje jeden element Stosu.

Metody publiczne

- [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.

Atrybuty prywatne

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

4.9.1 Opis szczegółowy

```
template<class typ>class Stos< typ >
```

Modeluje pojęcie Stosu.

Definicja w linii 19 pliku Stos.cpp.

4.9.2 Dokumentacja konstruktora i destruktora

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

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

Definicja w linii 85 pliku Stos.cpp.

4.9.3 Dokumentacja funkcji składowych

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

Usuwa 'górny' element Stosu

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 148 pliku Stos.cpp.

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

Dodaje daną podaną jako argument wywołania

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 129 pliku Stos.cpp.

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

Sprawdza ile aktualnie elementów znajduje się na Stosie

Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 174 pliku Stos.cpp.

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 200 pliku Stos.cpp.

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

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

Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 186 pliku Stos.cpp.

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

Zwalnia zaalokowana przez [Stos](#) pamięć

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez [Stos](#)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 109 pliku Stos.cpp.

4.9.4 Dokumentacja atrybutów składowych

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

Wskaźnik na pierwszy element Stosu

Definicja w linii 65 pliku Stos.cpp.

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

Przechowuje aktualną ilość Elementów znajdujących się na Stosie

Definicja w linii 73 pliku Stos.cpp.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Stos.cpp](#)

5 Dokumentacja plików

5.1 Dokumentacja pliku Framework.hh

Definicja klasy [Framework](#).

```
#include <iostream>
```

Komponenty

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

5.1.1 Opis szczegółowy

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

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

5.2 Dokumentacja pliku InterfejsADT.hh

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

Komponenty

- class [InterfejsADT< typ >](#)

5.3 Dokumentacja pliku Kolejka.cpp

Definicja klasy [Kolejka](#).

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

Komponenty

- class [Kolejka< typ >](#)

- Modeluje pojęcie Kolejki.*
- struct `Kolejka< typ >::Element`
Modeluje jeden element Kolejki.

Definicje

- #define `KOLEJKA_HH`

5.3.1 Opis szczegółowy

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

Definicja w pliku `Kolejka.cpp`.

5.3.2 Dokumentacja definicji

5.3.2.1 #define KOLEJKA_HH

Definicja w linii 2 pliku `Kolejka.cpp`.

5.4 Dokumentacja pliku Lista.cpp

Definicja klasy `Lista`.

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

Komponenty

- class `Lista< typ >`
Modeluje pojęcie listy.
- struct `Lista< typ >::Element`
Modeluje jeden element Listy.

Definicje

- #define `LISTA_HH`

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.cpp`.

5.4.2 Dokumentacja definicji

5.4.2.1 #define LISTA_HH

Definicja w linii 2 pliku `Lista.cpp`.

5.5 Dokumentacja pliku main.cpp

Moduł główny programu.

```
#include "../src/Lista.cpp"
#include "../src/Stos.cpp"
#include "../src/Kolejka.cpp"
#include "../inc/Statystyka.hh"
#include <ctime>
```

Definicje

- `#define ILOSC_POWTORZEN 10`
Ilość powtórzeń danej próby.
- `#define ILOSC_PROB 5`
Ilość prób.

Funkcje

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

5.5.1 Opis szczegółowy

Program wykonuje serię 10 pomiarów czasu wykonania metody start dla różnych wielkości problemu obliczeniowego, dla każdego zaimplementowanego typu danych - [Lista](#), [Stos](#), [Kolejka](#). 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](#), [Kolejka](#) lub [Stos](#)

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

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

5.5.2 Dokumentacja definicji

5.5.2.1 `#define ILOSC_POWTORZEN 10`

Ilość powtórzeń danej próby

Definicja w linii 33 pliku main.cpp.

5.5.2.2 `#define ILOSC_PROB 5`

Ilość prób = ilość rozmiarów prób

Definicja w linii 41 pliku main.cpp.

5.5.3 Dokumentacja funkcji

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

Definicja w linii 43 pliku main.cpp.

5.6 Dokumentacja pliku Pliki.cpp

Definicje funkcji obsługi plików.

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

Funkcje

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

5.6.1 Opis szczegółowy

Plik zawiera definicje funkcji związanych z obsługa plików.

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

5.6.2 Dokumentacja funkcji

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

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

Parametry

in	n	- ilość liczb do zapisania
in	zakres	- górny zakres wartości liczb

Definicja w linii 19 pliku Pliki.cpp.

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

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

Parametry

in	nazwaPliku	- nazwa pliku który chcemy otworzyć
in	plik	- strumień powiązany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.7 Dokumentacja pliku Pliki.hh

Funkcje obsługi plików.

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

Funkcje

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

5.7.1 Opis szczegółowy

Plik zawiera deklaracje funkcji związanych z obsługą plików

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

5.7.2 Dokumentacja funkcji

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

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

Parametry

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

Definicja w linii 19 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.

5.8 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy [Statystyka](#).

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

5.8.1 Opis szczegółowy

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

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

5.9 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy [Statystyka](#).

```
#include <iostream>
```

Komponenty

- class [Statystyka](#)

Modeluje pojęcie statystyki.

5.9.1 Opis szczegółowy

Zawiera definicję klasy [Statystyka](#)

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

5.10 Dokumentacja pliku Stos.cpp

Zawiera definicję Stosu.

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

Komponenty

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

5.10.1 Opis szczegółowy

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

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

Skorowidz

~Statystyka

Statystyka, 15

Czas

Statystyka, 16

Element

Kolejka::Element, 4

Lista::Element, 6

Stos::Element, 5

Framework, 6

Start, 7

WczytajDane, 7

Framework.hh, 19

ILOSC_POWTORZEN

main.cpp, 21

ILOSC_PROB

main.cpp, 21

IleProb

Statystyka, 16

InterfejsADT

pop, 8

push, 8

size, 8

Start, 8

WczytajDane, 8

Zwolnij, 9

InterfejsADT< typ >, 7

InterfejsADT.hh, 19

KOLEJKA_HH

Kolejka.cpp, 20

Kolejka

Kolejka, 10

Koniec, 11

Poczatek, 11

pop, 10

push, 10

Rozmiar, 11

size, 10

Start, 11

WczytajDane, 11

Zwolnij, 11

Kolejka< typ >, 9

Kolejka< typ >::Element, 3

Kolejka.cpp, 19

KOLEJKA_HH, 20

Kolejka::Element

Element, 4

nastepny, 4

wartosc, 4

Koniec

Kolejka, 11

Lista, 14

LISTA_HH

Lista.cpp, 20

Lista

Koniec, 14

Lista, 13

Poczatek, 14

pop, 13

push, 13

Rozmiar, 14

size, 13

Start, 13

WczytajDane, 14

Zwolnij, 14

Lista< typ >, 12

Lista< typ >::Element, 5

Lista.cpp, 20

LISTA_HH, 20

Lista::Element

Element, 6

nastepny, 6

wartosc, 6

LosujIntDoPliku

Pliki.cpp, 22

Pliki.hh, 23

main

main.cpp, 21

main.cpp, 21

ILOSC_POWTORZEN, 21

ILOSC_PROB, 21

main, 21

nastepny

Kolejka::Element, 4

Lista::Element, 6

Stos::Element, 5

OtworzPlikIn

Pliki.cpp, 22

Pliki.hh, 23

Pliki.cpp, 21

LosujIntDoPliku, 22

OtworzPlikIn, 22

Pliki.hh, 22

LosujIntDoPliku, 23

OtworzPlikIn, 23

Poczatek

Kolejka, 11

Lista, 14

Stos, 18

pop

InterfejsADT, 8

Kolejka, 10

Lista, 13

Stos, 17

Proba
 Statystyka, 16

push
 InterfejsADT, 8
 Kolejka, 10
 Lista, 13
 Stos, 17

Rozmiar
 Kolejka, 11
 Lista, 14
 Stos, 19

size
 InterfejsADT, 8
 Kolejka, 10
 Lista, 13
 Stos, 18

Start
 Framework, 7
 InterfejsADT, 8
 Kolejka, 11
 Lista, 13
 Stos, 18

Statystyka, 14
 ~Statystyka, 15
 Czas, 16
 IleProb, 16
 Proba, 16
 Statystyka, 15
 ZapiszStaty, 16

Statystyka.cpp, 23

Statystyka.hh, 23

Stos
 Poczatek, 18
 pop, 17
 push, 17
 Rozmiar, 19
 size, 18
 Start, 18
 Stos, 17
 WczytajDane, 18
 Zwolnij, 18

Stos< typ >, 16

Stos< typ >::Element, 4

Stos.cpp, 24

Stos::Element
 Element, 5
 nastepny, 5
 wartosc, 5

wartosc
 Kolejka::Element, 4
 Lista::Element, 6
 Stos::Element, 5

WczytajDane
 Framework, 7
 InterfejsADT, 8
 Kolejka, 11

Lista, 14

Stos, 18

ZapiszStaty
 Statystyka, 16

Zwolnij
 InterfejsADT, 9
 Kolejka, 11
 Lista, 14
 Stos, 18