

PAMSI\_LAB

Wygenerowano przez Doxygen 1.8.6

Cz, 26 mar 2015 08:59:00

## Spis treści

<b>1 Indeks hierarchiczny</b>	<b>1</b>
1.1 Hierarchia klas . . . . .	1
<b>2 Indeks klas</b>	<b>1</b>
2.1 Lista klas . . . . .	1
<b>3 Indeks plików</b>	<b>2</b>
3.1 Lista plików . . . . .	2
<b>4 Dokumentacja klas</b>	<b>2</b>
4.1 Dokumentacja szablonu klasy Benchmark< typ > . . . . .	2
4.1.1 Opis szczegółowy . . . . .	3
4.1.2 Dokumentacja konstruktora i destruktora . . . . .	3
4.1.3 Dokumentacja funkcji składowych . . . . .	3
4.1.4 Dokumentacja atrybutów składowych . . . . .	4
4.2 Dokumentacja struktury Kolejka< typ >::Element . . . . .	4
4.2.1 Opis szczegółowy . . . . .	4
4.2.2 Dokumentacja konstruktora i destruktora . . . . .	4
4.2.3 Dokumentacja atrybutów składowych . . . . .	6
4.3 Dokumentacja struktury Stos< typ >::Element . . . . .	6
4.3.1 Opis szczegółowy . . . . .	6
4.3.2 Dokumentacja konstruktora i destruktora . . . . .	6
4.3.3 Dokumentacja atrybutów składowych . . . . .	7
4.4 Dokumentacja struktury Lista< typ >::Element . . . . .	7
4.4.1 Opis szczegółowy . . . . .	7
4.4.2 Dokumentacja konstruktora i destruktora . . . . .	7
4.4.3 Dokumentacja atrybutów składowych . . . . .	8
4.5 Dokumentacja klasy Framework . . . . .	8
4.5.1 Opis szczegółowy . . . . .	8
4.5.2 Dokumentacja funkcji składowych . . . . .	9
4.6 Dokumentacja szablonu klasy InterfejsADT< typ > . . . . .	10
4.6.1 Opis szczegółowy . . . . .	11
4.6.2 Dokumentacja funkcji składowych . . . . .	11
4.7 Dokumentacja szablonu klasy Kolejka< 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 szablonu klasy Lista< typ > . . . . .	15

4.8.1	Opis szczegółowy . . . . .	16
4.8.2	Dokumentacja konstruktora i destruktora . . . . .	16
4.8.3	Dokumentacja funkcji składowych . . . . .	16
4.8.4	Dokumentacja atrybutów składowych . . . . .	18
4.9	Dokumentacja klasy Statystyka . . . . .	19
4.9.1	Opis szczegółowy . . . . .	19
4.9.2	Dokumentacja konstruktora i destruktora . . . . .	19
4.9.3	Dokumentacja funkcji składowych . . . . .	20
4.9.4	Dokumentacja atrybutów składowych . . . . .	20
4.10	Dokumentacja szablonu klasy Stos< typ > . . . . .	20
4.10.1	Opis szczegółowy . . . . .	21
4.10.2	Dokumentacja konstruktora i destruktora . . . . .	21
4.10.3	Dokumentacja funkcji składowych . . . . .	22
4.10.4	Dokumentacja atrybutów składowych . . . . .	23
<b>5</b>	<b>Dokumentacja plików</b>	<b>23</b>
5.1	Dokumentacja pliku Benchmark.hh . . . . .	23
5.1.1	Opis szczegółowy . . . . .	23
5.2	Dokumentacja pliku Framework.hh . . . . .	23
5.2.1	Opis szczegółowy . . . . .	24
5.3	Dokumentacja pliku InterfejsADT.hh . . . . .	24
5.4	Dokumentacja pliku Kolejka.hh . . . . .	24
5.4.1	Opis szczegółowy . . . . .	24
5.5	Dokumentacja pliku Lista.hh . . . . .	24
5.5.1	Opis szczegółowy . . . . .	25
5.6	Dokumentacja pliku main.cpp . . . . .	25
5.6.1	Opis szczegółowy . . . . .	25
5.6.2	Dokumentacja definicji . . . . .	25
5.6.3	Dokumentacja funkcji . . . . .	26
5.7	Dokumentacja pliku Pliki.cpp . . . . .	26
5.7.1	Opis szczegółowy . . . . .	26
5.7.2	Dokumentacja funkcji . . . . .	26
5.8	Dokumentacja pliku Pliki.hh . . . . .	27
5.8.1	Opis szczegółowy . . . . .	27
5.8.2	Dokumentacja funkcji . . . . .	27
5.9	Dokumentacja pliku Statystyka.cpp . . . . .	28
5.9.1	Opis szczegółowy . . . . .	28
5.10	Dokumentacja pliku Statystyka.hh . . . . .	28
5.10.1	Opis szczegółowy . . . . .	28
5.11	Dokumentacja pliku Stos.hh . . . . .	28

5.11.1 Opis szczegółowy . . . . .	29
<b>Indeks</b>	<b>30</b>

## 1 Indeks hierarchiczny

### 1.1 Hierarchia klas

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

<b>Benchmark&lt; typ &gt;</b>	<b>2</b>
<b>Kolejka&lt; typ &gt;::Element</b>	<b>4</b>
<b>Stos&lt; typ &gt;::Element</b>	<b>6</b>
<b>Lista&lt; typ &gt;::Element</b>	<b>7</b>
<b>Framework</b>	<b>8</b>
<b>InterfejsADT&lt; typ &gt;</b>	<b>10</b>
<b>Kolejka&lt; typ &gt;</b>	<b>12</b>
<b>Lista&lt; typ &gt;</b>	<b>15</b>
<b>Stos&lt; typ &gt;</b>	<b>20</b>
<b>Statystyka</b>	<b>19</b>

## 2 Indeks klas

### 2.1 Lista klas

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

<b>Benchmark&lt; typ &gt;</b> Modeluje pojęcie Benchmarku	<b>2</b>
<b>Kolejka&lt; typ &gt;::Element</b> Modeluje jeden element Kolejki	<b>4</b>
<b>Stos&lt; typ &gt;::Element</b> Modeluje jeden element Stosu	<b>6</b>
<b>Lista&lt; typ &gt;::Element</b> Modeluje jeden element Listy	<b>7</b>
<b>Framework</b> Modeluje interfejs programu	<b>8</b>
<b>InterfejsADT&lt; typ &gt;</b>	<b>10</b>
<b>Kolejka&lt; typ &gt;</b> Modeluje pojęcie Kolejki	<b>12</b>

<a href="#">Lista&lt; typ &gt;</a>	
Modeluje pojęcie listy	15
<a href="#">Statystyka</a>	
Modeluje pojęcie statystyki	19
<a href="#">Stos&lt; typ &gt;</a>	
Modeluje pojęcie Stosu	20

## 3 Indeks plików

### 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<a href="#">Benchmark.hh</a>	
Definicja klasy <a href="#">Benchmark</a>	23
<a href="#">Framework.hh</a>	
Definicja klasy <a href="#">Framework</a>	23
<a href="#">InterfejsADT.hh</a>	24
<a href="#">Kolejka.hh</a>	
Definicja klasy <a href="#">Kolejka</a>	24
<a href="#">Lista.hh</a>	
Definicja klasy <a href="#">Lista</a>	24
<a href="#">main.cpp</a>	
Moduł główny programu	25
<a href="#">Pliki.cpp</a>	
Definicje funkcji obsługi plików	26
<a href="#">Pliki.hh</a>	
Funkcje obsługi plików	27
<a href="#">Statystyka.cpp</a>	
Zawiera definicję metod klasy <a href="#">Statystyka</a>	28
<a href="#">Statystyka.hh</a>	
Zawiera definicję klasy <a href="#">Statystyka</a>	28
<a href="#">Stos.hh</a>	
Zawiera definicję Stosu	28

## 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 \*ileDanych, unsigned int ilePowtorzen)  
*Konstruktor 2 argumentowy.*
- void **Test** (Framework \*I, std::string nazwaPliku)  
*Testowanie algorytmu.*

## Atrybuty prywatne

- **Statystyka** \* stat  
*Statystyki testu.*
- unsigned int **ileProb**  
*Ilość prób.*
- unsigned int \* **ileDanych**  
*Tablica licznosci 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 * ileDanych, unsigned int ilePowtorzen ) [inline]`

Tworzy obiekt klasy **Benchmark** i inicjuje nową statystykę dla obiektu

## Parametry

in	<i>ileProb</i>	- ilość prób, które zostaną wykonane
in	<i>ileDanych</i>	- wskaźnik na tablice z licznosciami kolejnych serii
in	<i>ilePowtorzen</i>	- ilość powtórzeń każdej serii

Definicja w linii 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 nazwaPliku ) [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 kojenych 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 `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.*

#### 4.2.1 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.hh.

#### 4.2.2 Dokumentacja konstruktora i destruktor

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

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



**Parametry**

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

Definicja w linii 60 pliku Kolejka.hh.

**4.2.3 Dokumentacja atrybutów składowych****4.2.3.1 `template<class typ> Element* Kolejka< typ >::Element::nastepny`**

Wskaźnik na kolejny [Element](#) Kolejki

Definicja w linii 49 pliku Kolejka.hh.

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

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

Definicja w linii 42 pliku Kolejka.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Kolejka.hh](#)

**4.3 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.*

**4.3.1 Opis szczegółowy**

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

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

Definicja w linii 30 pliku Stos.hh.

**4.3.2 Dokumentacja konstruktora i destruktora****4.3.2.1 `template<class typ> Stos< typ >::Element::Element ( typ k ) [inline]`**

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

## Parametry

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

Definicja w linii 56 pliku Stos.hh.

## 4.3.3 Dokumentacja atrybutów składowych

## 4.3.3.1 template&lt;class typ&gt; Element\* Stos&lt; typ &gt;::Element::nastepny

Wskaźnik na kolejny [Element](#) Stosu

Definicja w linii 45 pliku Stos.hh.

## 4.3.3.2 template&lt;class typ&gt; typ Stos&lt; typ &gt;::Element::wartosc

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

Definicja w linii 38 pliku Stos.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Stos.hh](#)

## 4.4 Dokumentacja struktury Lista&lt; typ &gt;::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.*

## 4.4.1 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.hh.

## 4.4.2 Dokumentacja konstruktora i destruktor

## 4.4.2.1 template&lt;class typ&gt; Lista&lt; typ &gt;::Element::Element ( typ k ) [inline]

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

### Parametry

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

Definicja w linii 60 pliku Lista.hh.

### 4.4.3 Dokumentacja atrybutów składowych

#### 4.4.3.1 `template<class typ> Element* Lista< typ >::Element::nastepny`

Wskaźnik na kolejny [Element](#) Listy

Definicja w linii 49 pliku Lista.hh.

#### 4.4.3.2 `template<class typ> typ Lista< typ >::Element::wartosc`

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

Definicja w linii 42 pliku Lista.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

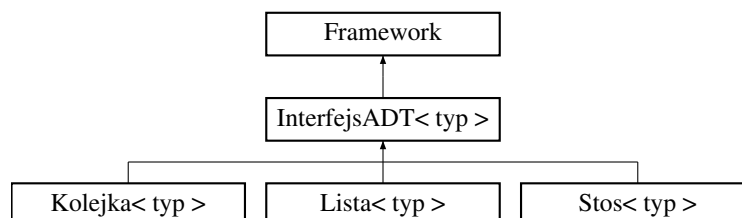
- [Lista.hh](#)

## 4.5 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.*

### 4.5.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanych w ramach kursu.

Definicja w linii 24 pliku Framework.hh.

#### 4.5.2 Dokumentacja funkcji składowych

##### 4.5.2.1 `virtual void Framework::Start ( const unsigned int k ) [pure virtual]`

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

**Parametry**

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

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

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

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

**Parametry**

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

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

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

Zwalnia pamięć zajmowaną przez obiekty wykorzystane do testów

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

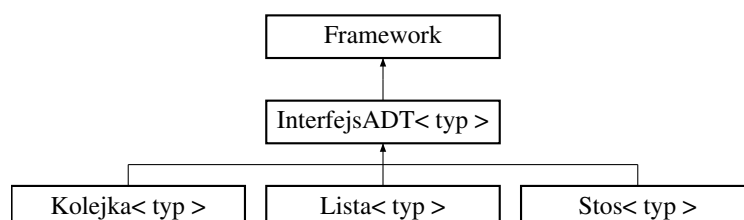
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Framework.hh](#)

## 4.6 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*

## 4.6.1 Opis szczegółowy

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

\ brief Definiuje interfejs użytkownika

Definiuje interfejs użytkownika dla listy, stosu i kolejki.

Definicja w linii 13 pliku InterfejsADT.hh.

## 4.6.2 Dokumentacja funkcji składowych

```
4.6.2.1 template<class typ > virtual void InterfejsADT< typ >::pop ( 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 [Lista< typ >](#), [Kolejka< typ >](#) i [Stos< typ >](#).

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

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

```
4.6.2.3 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 >](#).

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

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

Parametry

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

Implementuje [Framework](#).

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

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

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

## Parametry

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 >](#).

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

Zwalnia pamięć zajmowaną przez daną strukturę

Implementuje [Framework](#).

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

Dokumentacja dla tej klasy została wygenerowana z pliku:

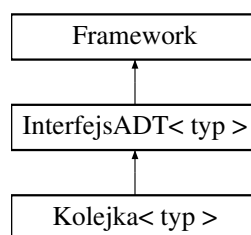
- [InterfejsADT.hh](#)

## 4.7 Dokumentacja szablonu klasy Kolejka< typ >

Modeluje pojęcie Kolejki.

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

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

## 4.7.1 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.hh.

## 4.7.2 Dokumentacja konstruktora i destruktor

```
4.7.2.1 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.hh.

## 4.7.3 Dokumentacja funkcji składowych

```
4.7.3.1 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 ustawoć 0!

## Parametry

in	<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.hh.

```
4.7.3.2 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

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 146 pliku Kolejka.hh.

```
4.7.3.3 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.hh.

**4.7.3.4** `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 Kolejkę 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.hh.

**4.7.3.5** `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.hh.

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

Zwalnia zaalokowaną przez Kolejke pamięć

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez Kolejkę

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 124 pliku Kolejka.hh.

**4.7.4 Dokumentacja atrybutów składowych**

**4.7.4.1** `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.hh.

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

Wskaźnik na pierwszy element Kolejki

Definicja w linii 72 pliku Kolejka.hh.

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

Przechowuje aktualną ilość Elementów znajdujących się w Kolejce

Definicja w linii 88 pliku Kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

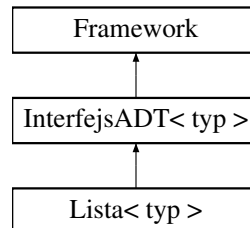
- [Kolejka.hh](#)

## 4.8 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](#) (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=0)  
*Wczytuje dane z pliku.*
- typ [operator\[\]](#) (size\_t pole) const  
*Wyciąga wartość elementu Listy.*
- 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.*

#### 4.8.1 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.hh.

#### 4.8.2 Dokumentacja konstruktora i destruktor

```
4.8.2.1 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.hh.

#### 4.8.3 Dokumentacja funkcji składowych

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

Wyluskuje wartość danego elementu z Listy

Parametry

in	<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 276 pliku Lista.hh.

```
4.8.3.2 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.hh.

```
4.8.3.3 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 ( <a href="#">siese()</a> jeżeli na koniec)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 143 pliku Lista.hh.

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

Sprawdza ile aktualnie elementów znajduje się na Liście

## Zwracane wartości

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 232 pliku Lista.hh.

**4.8.3.5** `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 obliczeniowa
-----------	----------	--

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 298 pliku Lista.hh.

**4.8.3.6** `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 246 pliku Lista.hh.

**4.8.3.7** `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 123 pliku Lista.hh.

## 4.8.4 Dokumentacja atrybutów składowych

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

Wskaźnik na ostatni element listy

Definicja w linii 80 pliku Lista.hh.

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

Wskaźnik na pierwszy element Listy

Definicja w linii 72 pliku Lista.hh.

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

Przechowuje aktualną ilość Elementów znajdujących się na Liście

Definicja w linii 87 pliku Lista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Lista.hh](#)

## 4.9 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.*

#### 4.9.1 Opis szczegółowy

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

Definicja w linii 22 pliku Statystyka.hh.

#### 4.9.2 Dokumentacja konstruktora i destruktora

##### 4.9.2.1 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.

##### 4.9.2.2 Statystyka::~~Statystyka ( ) [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablice przechowujące statystykę.

Definicja w linii 68 pliku Statystyka.hh.

### 4.9.3 Dokumentacja funkcji składowych

#### 4.9.3.1 `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.

#### 4.9.3.2 `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.

### 4.9.4 Dokumentacja atrybutów składowych

#### 4.9.4.1 `double* Statystyka::Czas [private]`

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 46 pliku Statystyka.hh.

#### 4.9.4.2 `unsigned int Statystyka::IleProb [private]`

Ilość prób do utworzenia statystyki

Definicja w linii 30 pliku Statystyka.hh.

#### 4.9.4.3 `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:

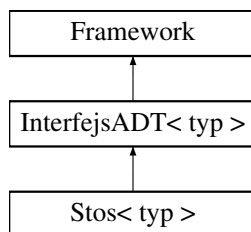
- [Statystyka.hh](#)
- [Statystyka.cpp](#)

## 4.10 Dokumentacja szablonu klasy Stos< typ >

Modeluje pojęcie Stosu.

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

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.*

#### 4.10.1 Opis szczegółowy

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

Modeluje pojęcie Stosu.

Definicja w linii 22 pliku Stos.hh.

#### 4.10.2 Dokumentacja konstruktora i destruktor

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

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

Definicja w linii 88 pliku Stos.hh.



## 4.10.3 Dokumentacja funkcji składowych

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

Usuwa 'górny' element Stosu

## Parametry

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

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 151 pliku Stos.hh.

4.10.3.2 `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 132 pliku Stos.hh.

4.10.3.3 `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 177 pliku Stos.hh.

4.10.3.4 `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 wypadku 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 203 pliku Stos.hh.

4.10.3.5 `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 189 pliku Stos.hh.

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

Zwalnia zaalokowaną przez [Stos](#) pamięć

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez [Stos](#)

Implementuje [InterfejsADT< typ >](#).

Definicja w linii 112 pliku [Stos.hh](#).

#### 4.10.4 Dokumentacja atrybutów składowych

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

Wskaźnik na pierwszy element Stosu

Definicja w linii 68 pliku [Stos.hh](#).

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

Przechowuje aktualną ilość Elementów znajdujących się na Stosie

Definicja w linii 76 pliku [Stos.hh](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Stos.hh](#)

## 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 Kolejka.hh

Definicja klasy [Kolejka](#).

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

## Komponenty

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

### 5.4.1 Opis szczegółowy

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

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

## 5.5 Dokumentacja pliku Lista.hh

Definicja klasy [Lista](#).

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

## Komponenty

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

### 5.5.1 Opis szczegółowy

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

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

## 5.6 Dokumentacja pliku main.cpp

Moduł główny programu.

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

## Definicje

- #define [ILOSC\\_POWTORZEN](#) 10  
*Ilość powtórzeń danej próby.*
- #define [ILOSC\\_PROB](#) 3  
*Ilość prób.*

## Funkcje

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

### 5.6.1 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 "TypDanych.dat". gdzie "TypDanych" to odpowiednio [Lista](#), [Kolejka](#) lub [Stos](#)

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

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

### 5.6.2 Dokumentacja definicji

#### 5.6.2.1 #define [ILOSC\\_POWTORZEN](#) 10

Ilość powtórzeń danej próby

Definicja w linii 34 pliku main.cpp.

### 5.6.2.2 #define ILOSC\_PROB 3

Ilość prób = ilość rozmiarów prób

Definicja w linii 42 pliku main.cpp.

### 5.6.3 Dokumentacja funkcji

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

Definicja w linii 44 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	n	- ilość liczb do zapisania
in	zakres	- 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](#).

## 5.11 Dokumentacja pliku Stos.hh

Zawiera definicję Stosu.

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

### Komponenty

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

#### 5.11.1 Opis szczegółowy

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

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



## Skorowidz

~Statystyka

Statystyka, 19

Benchmark

Benchmark, 3

IleDanych, 4

IlePowtorzen, 4

IleProb, 4

stat, 4

Test, 3

Benchmark< typ >, 2

Benchmark.hh, 23

Czas

Statystyka, 20

Element

Kolejka::Element, 4

Lista::Element, 7

Stos::Element, 6

Framework, 8

Start, 9

WczytajDane, 10

Zwolnij, 10

Framework.hh, 23

ILOSC\_POWTORZEN

main.cpp, 25

ILOSC\_PROB

main.cpp, 25

IleDanych

Benchmark, 4

IlePowtorzen

Benchmark, 4

IleProb

Benchmark, 4

Statystyka, 20

InterfejsADT

pop, 11

push, 11

size, 11

Start, 11

WczytajDane, 11

Zwolnij, 12

InterfejsADT< typ >, 10

InterfejsADT.hh, 24

Kolejka

Kolejka, 13

Koniec, 14

Poczatek, 14

pop, 13

push, 13

Rozmiar, 14

size, 13

Start, 14

WczytajDane, 14

Zwolnij, 14

Kolejka< typ >, 12

Kolejka< typ >::Element, 4

Kolejka.hh, 24

Kolejka::Element

Element, 4

nastepny, 6

wartosc, 6

Koniec

Kolejka, 14

Lista, 18

Lista

Koniec, 18

Lista, 16

Poczatek, 18

pop, 16

push, 16

Rozmiar, 18

size, 16

Start, 18

WczytajDane, 18

Zwolnij, 18

Lista< typ >, 15

Lista< typ >::Element, 7

Lista.hh, 24

Lista::Element

Element, 7

nastepny, 8

wartosc, 8

LosujIntDoPliku

Pliki.cpp, 26

Pliki.hh, 27

main

main.cpp, 26

main.cpp, 25

ILOSC\_POWTORZEN, 25

ILOSC\_PROB, 25

main, 26

nastepny

Kolejka::Element, 6

Lista::Element, 8

Stos::Element, 7

OtworzPlikIn

Pliki.cpp, 26

Pliki.hh, 27

OtworzPlikOut

Pliki.cpp, 27

Pliki.hh, 28

Pliki.cpp, 26

LosujIntDoPliku, 26

- OtworzPlikIn, [26](#)
- OtworzPlikOut, [27](#)
- Pliki.hh, [27](#)
  - LosujIntDoPliku, [27](#)
  - OtworzPlikIn, [27](#)
  - OtworzPlikOut, [28](#)
- Poczatek
  - Kolejka, [14](#)
  - Lista, [18](#)
  - Stos, [23](#)
- pop
  - InterfejsADT, [11](#)
  - Kolejka, [13](#)
  - Lista, [16](#)
  - Stos, [22](#)
- Proba
  - Statystyka, [20](#)
- push
  - InterfejsADT, [11](#)
  - Kolejka, [13](#)
  - Lista, [16](#)
  - Stos, [22](#)
- Rozmiar
  - Kolejka, [14](#)
  - Lista, [18](#)
  - Stos, [23](#)
- size
  - InterfejsADT, [11](#)
  - Kolejka, [13](#)
  - Lista, [16](#)
  - Stos, [22](#)
- Start
  - Framework, [9](#)
  - InterfejsADT, [11](#)
  - Kolejka, [14](#)
  - Lista, [18](#)
  - Stos, [22](#)
- stat
  - Benchmark, [4](#)
- Statystyka, [19](#)
  - ~Statystyka, [19](#)
  - Czas, [20](#)
  - IleProb, [20](#)
  - Proba, [20](#)
  - Statystyka, [19](#)
  - ZapiszStaty, [20](#)
- Statystyka.cpp, [28](#)
- Statystyka.hh, [28](#)
- Stos
  - Poczatek, [23](#)
  - pop, [22](#)
  - push, [22](#)
  - Rozmiar, [23](#)
  - size, [22](#)
  - Start, [22](#)
  - Stos, [21](#)
  - WczytajDane, [22](#)
  - Zwolnij, [22](#)
- Stos< typ >, [20](#)
- Stos< typ >::Element, [6](#)
- Stos.hh, [28](#)
- Stos::Element
  - Element, [6](#)
  - nastepny, [7](#)
  - wartosc, [7](#)
- Test
  - Benchmark, [3](#)
- wartosc
  - Kolejka::Element, [6](#)
  - Lista::Element, [8](#)
  - Stos::Element, [7](#)
- WczytajDane
  - Framework, [10](#)
  - InterfejsADT, [11](#)
  - Kolejka, [14](#)
  - Lista, [18](#)
  - Stos, [22](#)
- ZapiszStaty
  - Statystyka, [20](#)
- Zwolnij
  - Framework, [10](#)
  - InterfejsADT, [12](#)
  - Kolejka, [14](#)
  - Lista, [18](#)
  - Stos, [22](#)