

PAMSI LAB II

0.3

Wygenerowano przez Doxygen 1.8.6

Śr, 25 mar 2015 22:00:02

Spis treści

1	Strona główna	1
1.1	Programu	1
2	Indeks hierarchiczny	3
2.1	Hierarchia klas	3
3	Indeks klas	5
3.1	Lista klas	5
4	Indeks plików	7
4.1	Lista plików	7
5	Dokumentacja klas	9
5.1	Dokumentacja klasy BenchmarkInterfejs	9
5.1.1	Opis szczegółowy	9
5.1.2	Dokumentacja funkcji składowych	9
5.1.2.1	_Generator	9
5.1.2.2	_Test	10
5.1.2.3	_Wczytaj	10
5.1.2.4	_WykonajTest	10
5.1.2.5	_Zwolnij	10
5.2	Dokumentacja klasy Kolejka	11
5.2.1	Opis szczegółowy	11
5.2.2	Dokumentacja konstruktora i destruktora	12
5.2.2.1	Kolejka	12
5.2.2.2	~Kolejka	12
5.2.3	Dokumentacja funkcji składowych	12
5.2.3.1	_Pokaz	12
5.2.3.2	_Pop	12
5.2.3.3	_Push	12
5.2.3.4	_Rozmiar	12
5.2.3.5	_Zwolnij	13
5.2.4	Dokumentacja atrybutów składowych	13

5.2.4.1	_Ilosc	13
5.2.4.2	_Ostatni	13
5.2.4.3	_Pierwszy	13
5.3	Dokumentacja klasy Lista	13
5.3.1	Opis szczegółowy	14
5.3.2	Dokumentacja konstruktora i destruktora	14
5.3.2.1	Lista	14
5.3.2.2	~Lista	14
5.3.3	Dokumentacja funkcji składowych	14
5.3.3.1	_Pokaz	14
5.3.3.2	_Pop	15
5.3.3.3	_Push	15
5.3.3.4	_Rozmiar	15
5.3.3.5	_Zwolnij	15
5.3.4	Dokumentacja atrybutów składowych	15
5.3.4.1	_Ilosc	16
5.3.4.2	Glowa	16
5.4	Dokumentacja klasy ListaTab	16
5.4.1	Opis szczegółowy	16
5.4.2	Dokumentacja konstruktora i destruktora	17
5.4.2.1	ListaTab	17
5.4.2.2	~ListaTab	17
5.4.3	Dokumentacja funkcji składowych	17
5.4.3.1	_Pokaz	17
5.4.3.2	_Pop	17
5.4.3.3	_Push	17
5.4.3.4	_Rozmiar	18
5.4.3.5	_Zwolnij	18
5.4.4	Dokumentacja atrybutów składowych	18
5.4.4.1	_L	18
5.4.4.2	_RozmiarL	18
5.4.4.3	_RozmiarT	18
5.5	Dokumentacja klasy Stos	19
5.5.1	Opis szczegółowy	19
5.5.2	Dokumentacja konstruktora i destruktora	20
5.5.2.1	Stos	20
5.5.2.2	~Stos	20
5.5.2.3	Stos	20
5.5.3	Dokumentacja funkcji składowych	20
5.5.3.1	_Pokaz	20

5.5.3.2	_Pop	20
5.5.3.3	_Push	20
5.5.3.4	_Rozmiar	21
5.5.3.5	_Zwolnij	21
5.5.4	Dokumentacja atrybutów składowych	21
5.5.4.1	_Ilosc	21
5.5.4.2	Gora	21
5.6	Dokumentacja klasy StosTab	21
5.6.1	Opis szczegółowy	22
5.6.2	Dokumentacja konstruktora i destruktora	22
5.6.2.1	StosTab	22
5.6.2.2	~StosTab	22
5.6.3	Dokumentacja funkcji składowych	22
5.6.3.1	_Pokaz	22
5.6.3.2	_Pop	23
5.6.3.3	_Push	23
5.6.3.4	_Rozmiar	23
5.6.3.5	_Zwolnij	23
5.6.4	Dokumentacja atrybutów składowych	24
5.6.4.1	_L	24
5.6.4.2	_RozmiarL	24
5.6.4.3	_RozmiarT	24
5.7	Dokumentacja klasy Struktury	24
5.7.1	Opis szczegółowy	25
5.7.2	Dokumentacja funkcji składowych	25
5.7.2.1	_Pokaz	25
5.7.2.2	_Pop	25
5.7.2.3	_Push	25
5.7.2.4	_Rozmiar	25
5.7.2.5	_Zwolnij	26
5.8	Dokumentacja klasy StrukturyBenchmark	26
5.8.1	Opis szczegółowy	26
5.8.2	Dokumentacja konstruktora i destruktora	27
5.8.2.1	StrukturyBenchmark	27
5.8.2.2	~StrukturyBenchmark	27
5.8.3	Dokumentacja funkcji składowych	27
5.8.3.1	_Przydziel	27
5.8.3.2	_Test	27
5.8.3.3	_Ustaw	27
5.8.3.4	_Wczytaj	28

5.8.3.5	<code>_Zwolnij</code>	28
5.8.4	Dokumentacja atrybutów składowych	28
5.8.4.1	<code>S</code>	28
5.8.4.2	<code>W</code>	28
5.9	Dokumentacja klasy <code>StrukturyTablice</code>	28
5.9.1	Opis szczegółowy	29
5.9.2	Dokumentacja funkcji składowych	29
5.9.2.1	<code>_Pokaz</code>	29
5.9.2.2	<code>_Pop</code>	29
5.9.2.3	<code>_Push</code>	29
5.9.2.4	<code>_Rozmiar</code>	29
5.10	Dokumentacja klasy <code>TabListaPod</code>	29
5.10.1	Opis szczegółowy	30
5.10.2	Dokumentacja konstruktora i destruktora	30
5.10.2.1	<code>TabListaPod</code>	30
5.10.2.2	<code>TabListaPod</code>	30
5.10.2.3	<code>~TabListaPod</code>	30
5.10.3	Dokumentacja funkcji składowych	30
5.10.3.1	<code>_Pokaz</code>	30
5.10.3.2	<code>_Pop</code>	30
5.10.3.3	<code>_Push</code>	31
5.10.3.4	<code>_Rozmiar</code>	31
5.10.3.5	<code>_XRozszerz</code>	31
5.10.3.6	<code>_Zwolnij</code>	31
5.10.4	Dokumentacja atrybutów składowych	32
5.10.4.1	<code>_L</code>	32
5.10.4.2	<code>_RozmiarL</code>	32
5.10.4.3	<code>_RozmiarT</code>	32
5.11	Dokumentacja struktury <code>Stos::Wezel</code>	32
5.11.1	Opis szczegółowy	32
5.11.2	Dokumentacja atrybutów składowych	32
5.11.2.1	<code>_Nast</code>	32
5.11.2.2	<code>_Wartosc</code>	33
5.12	Dokumentacja struktury <code>Lista::Wezel</code>	33
5.12.1	Opis szczegółowy	33
5.12.2	Dokumentacja atrybutów składowych	33
5.12.2.1	<code>_Nast</code>	33
5.12.2.2	<code>_Wartosc</code>	33
5.13	Dokumentacja struktury <code>Kolejka::Wezel</code>	33
5.13.1	Opis szczegółowy	34

5.13.2 Dokumentacja atrybutów składowych	34
5.13.2.1 _Nast	34
5.13.2.2 _Wartosc	34
6 Dokumentacja plików	35
6.1 Dokumentacja pliku BenchmarkInterfejs.cpp	35
6.1.1 Opis szczegółowy	35
6.2 Dokumentacja pliku BenchmarkInterfejs.h	35
6.3 Dokumentacja pliku Kolejka.cpp	35
6.3.1 Opis szczegółowy	36
6.4 Dokumentacja pliku Kolejka.h	36
6.5 Dokumentacja pliku Lista.cpp	36
6.5.1 Opis szczegółowy	36
6.6 Dokumentacja pliku Lista.h	36
6.7 Dokumentacja pliku ListaTab.cpp	36
6.7.1 Opis szczegółowy	37
6.8 Dokumentacja pliku ListaTab.h	37
6.9 Dokumentacja pliku Main.cpp	37
6.9.1 Opis szczegółowy	37
6.9.2 Dokumentacja definicji	37
6.9.2.1 ILOSC_DANYCH	37
6.9.3 Dokumentacja funkcji	38
6.9.3.1 main	38
6.10 Dokumentacja pliku Stos.cpp	38
6.10.1 Opis szczegółowy	38
6.11 Dokumentacja pliku Stos.h	38
6.12 Dokumentacja pliku StosTab.cpp	38
6.12.1 Opis szczegółowy	38
6.13 Dokumentacja pliku StosTab.h	38
6.14 Dokumentacja pliku strona-glowna.dox	39
6.15 Dokumentacja pliku Struktury.h	39
6.16 Dokumentacja pliku StrukturyBenchmark.cpp	39
6.16.1 Opis szczegółowy	39
6.17 Dokumentacja pliku StrukturyBenchmark.h	39
6.18 Dokumentacja pliku StrukturyTablice.h	40
6.19 Dokumentacja pliku TabListaPod.cpp	40
6.19.1 Opis szczegółowy	40
6.20 Dokumentacja pliku TabListaPod.h	40
Indeks	41

Rozdział 1

Strona główna

Autor

Bartłomiej Ankowski

Data

19.03.2015

Wersja

0.3

1.1 Programu

Rozdział 2

Indeks hierarchiczny

2.1 Hierarchia klas

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

BenchmarkInterfejs	9
StrukturyBenchmark	26
Struktury	24
Kolejka	11
Lista	13
ListaTab	16
Stos	19
StosTab	21
TabListaPod	29
StrukturyTablice	28
Stos::Wezel	32
Lista::Wezel	33
Kolejka::Wezel	33

Rozdział 3

Indeks klas

3.1 Lista klas

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

BenchmarkInterfejs	
Modeluje pojecie Interfejsu Benchmark'u	9
Kolejka	
Modeluje pojecie Kolejki	11
Lista	13
ListaTab	16
Stos	
Modeluje pojecie Stosu	19
StosTab	21
Struktury	
Modeluje pojecie Struktury danych, klasa bazowa dla Stosu,Kolejki i Listy,zarowno w impleme- netacji wskaznikowej jak i tablicowej	24
StrukturyBenchmark	26
StrukturyTablice	28
TabListaPod	29
Stos::Wezel	32
Lista::Wezel	33
Kolejka::Wezel	33

Rozdział 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

BenchmarkInterfejs.cpp	
Definicje Metod klasy BenchmarkInterfejs	35
BenchmarkInterfejs.h	35
Kolejka.cpp	
Definicje Metod klasy Kolejka	35
Kolejka.h	36
Lista.cpp	
Definicje Metod klasy Lista	36
Lista.h	36
ListaTab.cpp	
Zawiera definicje Metod klasy ListaTab	36
ListaTab.h	37
Main.cpp	
Funkcja glowna programu	37
Stos.cpp	
Definicje metod klasy Stos	38
Stos.h	38
StosTab.cpp	
Zawiera definicje Metod klasy ListaTab	38
StosTab.h	38
Struktury.h	39
StrukturyBenchmark.cpp	
Definicje metod klasy StrukturyBenchmark	39
StrukturyBenchmark.h	39
StrukturyTablice.h	40
TabListaPod.cpp	
Definicje Metod Klasy TabListaPod	40
TabListaPod.h	40

Rozdział 5

Dokumentacja klas

5.1 Dokumentacja klasy BenchmarkInterfejs

Modeluje pojecie Interfejsu Benchmark'u.

```
#include <BenchmarkInterfejs.h>
```

Diagram dziedziczenia dla BenchmarkInterfejs

Metody publiczne

- virtual void `_Test` (const unsigned int Ilosc)=0
Metoda Wykonujaca pojedyncza operacje.
- virtual void `_Wczytaj` (string PlikIn, const unsigned n)=0
Metoda wczytujaca dane z pliku Metoda ma za zadanie wczytac dane z pliku wejscowego.
- void `_WykonajTest` (const unsigned int Ilosc_Pow)
Metoda wykonujaca test odpowiedniej struktury.
- void `_Generator` (string PlikOut, int n)
- virtual void `_Zwolnij` (const unsigned int n)=0
Metoda zwalnijaca pamiec.

5.1.1 Opis szczegółowy

Modeluje pojecie Interfejsu Benchmark'u.

Klasa bazowa dla implementowania benchmarku dla kolejnych struktur danych

Definicja w linii 27 pliku BenchmarkInterfejs.h.

5.1.2 Dokumentacja funkcji składowych

5.1.2.1 void BenchmarkInterfejs::_Generator (string PlikOut, int n)

Metoda generujaca dane

Metoda ma za zadanie wygenerowac plik z danymi z przedzialu (1-99), jest wywoływana gdy uzytkownik nie poda w argumencie wywołania programu nazwy pliku wejscowego z danymi

Parametry

in	<i>PlikOut</i>	- Nazwa plik w ktorym zapisywane sa wygenerowane dane
in	<i>n</i>	- Ilosc wygenerowanych danych

Definicja w linii 32 pliku BenchmarkInterfejs.cpp.

Oto graf wywoływań tej funkcji:

5.1.2.2 `virtual void BenchmarkInterfejs::_Test (const unsigned int llosc) [pure virtual]`

Metoda Wykonujaca pojedyncza operacje.

Metoda ma za zadanie wykonan pojedyncza operacja, ktorej czas jest rejestrowany

Parametry

in	<i>llosc</i>	- Liczba danych poddana testowi
----	--------------	---------------------------------

Implementowany w [StrukturyBenchmark](#).

5.1.2.3 `virtual void BenchmarkInterfejs::_Wczytaj (string PlikIn, const unsigned n) [pure virtual]`

Metoda wczytujaca dane z pliku Metoda ma za zadanie wczytac dane z pliku wejscowego.

Parametry

in	<i>PlikIn</i>	- Nazwa pliku wejscowego
in	<i>n</i>	- liczba wczytywanych danych

5.1.2.4 `void BenchmarkInterfejs::_WykonajTest (const unsigned int llosc_Pow)`

Metoda wykonujaca test odpowiedniej struktury.

Metoda ma za zadanie wykonac Benchmark dla struktury, dla ustawionej ilosci danych i okreslona przez argument metody ilosc powtorzen.

Parametry

in	<i>llosc_Pow</i>	- okresla ile razy ma sie wykonac test
----	------------------	--

Definicja w linii 11 pliku BenchmarkInterfejs.cpp.

Oto graf wywoływań tej funkcji:

5.1.2.5 `virtual void BenchmarkInterfejs::_Zwolnij (const unsigned int n) [pure virtual]`

Metoda zwalnijaca pamiec.

Metoda ma zazadanie wykonac operacje zwalania pamieci

Parametry

in	<i>n</i>	- liczba danych ktora bedzie zwolniona z pamieci
----	----------	--

Implementowany w [StrukturyBenchmark](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

- [BenchmarkInterfejs.h](#)
- [BenchmarkInterfejs.cpp](#)

5.2 Dokumentacja klasy Kolejka

Modeluje pojecie Kolejki.

```
#include <Kolejka.h>
```

Diagram dziedziczenia dla Kolejka

Diagram współpracy dla Kolejka:

Komponenty

- struct [Wezel](#)

Metody publiczne

- [Kolejka](#) ()
Konstruktor.
- [~Kolejka](#) ()
Destruktor.
- void [_Pokaz](#) ()
Metoda wyswietlajaca elementu Kolejki.
- void [_Push](#) (int k, unsigned int Pozycja=0)
Metoda dodajaca nowy [Wezel](#).
- int [_Pop](#) (unsigned int Pozycja=0)
Metda usuwajaca wezel.
- unsigned int [_Rozmiar](#) ()
Metoda informujaca o obecnej ilosci Wezlow.
- void [_Zwolnij](#) ()
Metoda zwalnijaca pamiec zajeta przez struktre.

Atrybuty prywatne

- [Wezel](#) * [_Pierwszy](#)
Pole klasy [Kolejka](#).
- [Wezel](#) * [_Ostatni](#)
Pole klasy [Kolejka](#).
- unsigned int [_Ilosc](#)
Pole klasy [Kolejka](#).

5.2.1 Opis szczegółowy

Modeluje pojecie Kolejki.

Klasa modeluje pojecie kolejki, dodajac nowy element na jej koniec i sciagajac pierwszy dodany element

Definicja w linii 22 pliku Kolejka.h.

5.2.2 Dokumentacja konstruktora i destruktora

5.2.2.1 Kolejka::Kolejka () [inline]

Konstruktor.

Ustawia wskaźniki na NULL oraz zeruje ilość elementów przy tworzeniu obiektów danej klasy

Definicja w linii 66 pliku Kolejka.h.

5.2.2.2 Kolejka::~~Kolejka ()

Destruktor.

Definicja w linii 66 pliku Kolejka.cpp.

Oto graf wywołań dla tej funkcji:

5.2.3 Dokumentacja funkcji składowych

5.2.3.1 void Kolejka::_Pokaz () [virtual]

Metoda wyświetlająca elementu Kolejki.

Metoda ma za zadanie wyświetlić wszystkie wartości znajdujące się w kolejce

Implementuje [Struktury](#).

Definicja w linii 51 pliku Kolejka.cpp.

5.2.3.2 int Kolejka::_Pop (unsigned int *Pozycja* = 0) [virtual]

Metoda usuwająca węzeł.

Metoda ma za zadanie usunąć pierwszy dodany element z kolejki oraz ustawić wskaźnik przed ostatnim elementem na NULL

Implementuje [Struktury](#).

Definicja w linii 38 pliku Kolejka.cpp.

Oto graf wywołań tej funkcji:

5.2.3.3 void Kolejka::_Push (int *k*, unsigned int *Pozycja* = 0) [virtual]

Metoda dodająca nowy [Węzeł](#).

Metoda ma za zadanie dodać nowy węzeł na koniec kolejki, umieścić w nim wartość zadana jako argument oraz pokazać wskaźnikiem na ostatni [Węzeł](#) przed dodaniem nowego Węzła

Parametry

<i>in</i>	<i>k</i>	- Wartość która zostanie umieszczona w odpowiednim polu Węzła
-----------	----------	---

Implementuje [Struktury](#).

Definicja w linii 19 pliku Kolejka.cpp.

5.2.3.4 unsigned int Kolejka::_Rozmiar () [inline], [virtual]

Metoda informująca o obecnej ilości Węzłów.

Metoda zwraca informacje o ilości obecnie znajdujących się elementów w kolejce

Zwraca

- Zwraca ilosc elementow w kolejce

Implementuje [Struktury](#).

Definicja w linii 102 pliku Kolejka.h.

5.2.3.5 void Kolejka::_Zwolnij() [virtual]

Metoda zwalnijajaca pamiec zajeta przez struktre.

Metoda ma za zadanie zwolnij pamiec zajeta przez zaladowane do struktury dane, elementy sa usuwany dopoki wskaznik pokazujacy na poczatek kolejki nie bedzie wskazywal na NULL

Implementuje [Struktury](#).

Definicja w linii 6 pliku Kolejka.cpp.

5.2.4 Dokumentacja atrybutów składowych**5.2.4.1 unsigned int Kolejka::_Ilosc [private]**

Pole klasy [Kolejka](#).

Pole przechowuje infromacje o ilosci obecnie znajdujacych sie elementow w kolejce

Definicja w linii 58 pliku Kolejka.h.

5.2.4.2 Wezel* Kolejka::_Ostatni [private]

Pole klasy [Kolejka](#).

Pole jest wskaznikiem na ostatni element kolejki

Definicja w linii 51 pliku Kolejka.h.

5.2.4.3 Wezel* Kolejka::_Pierwszy [private]

Pole klasy [Kolejka](#).

Pole jest wskaznikiem na pierwszy element kolejki

Definicja w linii 45 pliku Kolejka.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Kolejka.h](#)
- [Kolejka.cpp](#)

5.3 Dokumentacja klasy Lista

```
#include <Lista.h>
```

Diagram dziedziczenia dla Lista

Diagram współpracy dla Lista:

Komponenty

- struct [Wezel](#)

Metody publiczne

- void `_Push` (int wart, unsigned int poz=1)
Metoda dodajaca [Wezel](#).
- int `_Pop` (unsigned int Pozycja=1)
Metoda usuwajaca [Wezel](#).
- unsigned int `_Rozmiar` ()
Metoda informujaca o ilosci wezlow.
- [Lista](#) ()
Konstruktor.
- `~Lista` ()
Destruktor Usuwa wskaznik.
- void `_Pokaz` ()
Konstruktor Kopiujacy.
- void `_Zwolnij` ()
Metoda zwalnijaca pamiec zajeta przez struktre.

Atrybuty prywatne

- [Wezel](#) * [Glowa](#)
Pole klasy [Lista](#) Wskaznik na nowo dodany [Wezel](#).
- unsigned int `_Ilosc`
Pole klasy [Lista](#) Pole przechowuje bierzaca ilosc elementow listy.

5.3.1 Opis szczegółowy

Definicja w linii 12 pliku Lista.h.

5.3.2 Dokumentacja konstruktora i destruktora

5.3.2.1 `Lista::Lista ()` [inline]

Konstruktor.

Konstruktor ustawia wskaznik na NULL i zeruje ilosc wezlow listy

Definicja w linii 72 pliku Lista.h.

5.3.2.2 `Lista::~Lista ()` [inline]

Destruktor Usuwa wskaznik.

Definicja w linii 77 pliku Lista.h.

5.3.3 Dokumentacja funkcji składowych

5.3.3.1 `void Lista::_Pokaz ()` [virtual]

Konstruktor Kopiujacy.

Metoda wyswietlajaca elementy Listy

Metoda ma za zadanie wyswietlic wszystkie wartosci znajdujace sie na Liscie

Implementuje [Struktury](#).

Definicja w linii 90 pliku Lista.cpp.

5.3.3.2 `int Lista::_Pop (unsigned int Pozycja = 1) [virtual]`

Metoda usuwająca [Wezel](#).

Metoda ma za zadanie usunąć węzeł zgodny z argumentem metody [Wezel](#) zosatnie usunięty, a sąsiednie elementy zostaną połączone wskaźnikiem

Parametry

<code>in</code>	<i>Pozycja</i>	- Numer Węzła który zostanie usunięty
-----------------	----------------	---------------------------------------

Implementuje [Struktury](#).

Definicja w linii 22 pliku Lista.cpp.

5.3.3.3 `void Lista::_Push (int wart, unsigned int poz = 1) [virtual]`

Metoda dodająca [Wezel](#).

Metoda ma za zadanie dodać element do listy w zależności od argumentu miejscu i ustawić wartość zadana przez argument

Parametry

<code>in</code>	<i>wart</i>	- Wartość jaka zostanie dodana do Węzła
<code>in</code>	<i>poz</i>	- Pozycja w której zosatnie dodany Wezel

Implementuje [Struktury](#).

Definicja w linii 59 pliku Lista.cpp.

5.3.3.4 `unsigned int Lista::_Rozmiar () [inline],[virtual]`

Metoda informująca o ilości węzłów.

Metoda zwraca informacje o ilości aktualnych węzłów listy

Zwraca

- Ilość elementów listy

Implementuje [Struktury](#).

Definicja w linii 66 pliku Lista.h.

5.3.3.5 `void Lista::_Zwolnij () [virtual]`

Metoda zwalniania pamięć zajęta przez struktury.

Metoda ma za zadanie zwolnić pamięć zajęta przez załadowane do struktury dane, elementy są usuwane dopóki wskaźnik pokazujący na początek listy nie będzie wskazywał na NULL

Implementuje [Struktury](#).

Definicja w linii 9 pliku Lista.cpp.

5.3.4 Dokumentacja atrybutów składowych

5.3.4.1 unsigned int Lista::_Ilosc [private]

Pole klasy [Lista](#) Pole przechowuje bierzaca ilosc elementow listy.

Definicja w linii 39 pliku Lista.h.

5.3.4.2 Wezel* Lista::Glowa [private]

Pole klasy [Lista](#) Wskaznik na nowo dodany [Wezel](#).

Definicja w linii 34 pliku Lista.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Lista.h](#)
- [Lista.cpp](#)

5.4 Dokumentacja klasy ListaTab

```
#include <ListaTab.h>
```

Diagram dziedziczenia dla ListaTab

Diagram współpracy dla ListaTab:

Metody publiczne

- [ListaTab](#) ()
Konstruktor Podczas tworzenia obiektu tej klasy automatycznie alokowana jest tablica o rozmiarze 1 oraz ustawienie obecnej liczby elementow listy na 0.
- [~ListaTab](#) ()
Konstruktor Kopiujący.
- void [_Pokaz](#) ()
Metoda wypisujaca elementy listy.
- int [_Pop](#) (unsigned int Pozycja)
Metoda sciagajaca element z listy.
- void [_Push](#) (int k, unsigned int Pozycja)
Metoda dodajaca element do tablicy.
- unsigned int [_Rozmiar](#) ()
Metoda zwracajaca rozmiar listy.
- void [_Zwolnij](#) ()

Atrybuty prywatne

- int * [_L](#)
Pole klasy [ListaTab](#).
- unsigned int [_RozmiarL](#)
Pole Klasy [ListaTab](#).
- unsigned int [_RozmiarT](#)
Pole Klasy [ListaTab](#).

5.4.1 Opis szczegółowy

Definicja w linii 13 pliku ListaTab.h.

5.4.2 Dokumentacja konstruktora i destruktora

5.4.2.1 ListaTab::ListaTab ()

Konstruktor Podczas tworzenia obiektu tej klasy automatycznie alokowana jest tablica o rozmiarze 1 oraz ustawienie obecnej liczby elementów listy na 0.

Definicja w linii 65 pliku ListaTab.cpp.

5.4.2.2 ListaTab::~~ListaTab () [inline]

Konstruktor Kopiujacy.

Destruktor obiektu

Definicja w linii 48 pliku ListaTab.h.

5.4.3 Dokumentacja funkcji składowych

5.4.3.1 void ListaTab::_Pokaz () [virtual]

Metoda wypisująca elementy listy.

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

Implementuje [Struktury](#).

Definicja w linii 76 pliku ListaTab.cpp.

5.4.3.2 int ListaTab::_Pop (unsigned int *Pozycja*) [virtual]

Metoda ściągająca element z listy.

Metoda ma za zadanie ściągnąć wybrany przez użytkownika element listy oraz każdorazowo zmniejszyć tablicę z danymi o 1 oraz przekopiować pozostałe elementy listy

Parametry

in	<i>Pozycja</i>	- numer elementu który zostanie usunięty z listy i zostanie zwrócona jego wartość
----	----------------	---

Zwraca

Zwraca wybrany przez użytkownika element

Implementuje [Struktury](#).

Definicja w linii 13 pliku ListaTab.cpp.

5.4.3.3 void ListaTab::_Push (int *k*, unsigned int *Pozycja*) [virtual]

Metoda dodająca element do tablicy.

Metoda ma za zadanie dodać nowy element w wybranym przez użytkownika miejscu oraz zwiększyć każdorazowo tablicę danych o 1 i przepisać pozostałe elementy do nowej

Parametry

in	<i>k</i>	- wartosc jaka chcemy dodac do listy
in	<i>Pozycja</i>	- Pozycja na ktorej chcemy dodac wartosc

Implementuje [Struktury](#).

Definicja w linii 41 pliku ListaTab.cpp.

5.4.3.4 unsigned int ListaTab::_Rozmiar () [inline],[virtual]

Metoda zwracajaca rozmiar listy.

Metoda zwraca informacje o obecnej ilosci danych w strukturze

Zwraca

Zwraca ilosc elementow listy

Implementuje [Struktury](#).

Definicja w linii 87 pliku ListaTab.h.

5.4.3.5 void ListaTab::_Zwolnij () [virtual]

Metoda zwalnijajaca pamiec

Metoda ma za zadanie zwolnij pamiec zajmowana przez dane, dopoki ilosc elementow listy nie wynosi 0 wykonywana jest metoda _Pop, aby oproznic stos i zwolnic pamiec

Implementuje [Struktury](#).

Definicja w linii 7 pliku ListaTab.cpp.

5.4.4 Dokumentacja atrybutów składowych**5.4.4.1 int* ListaTab::_L [private]**

Pole klasy [ListaTab](#).

Pole zawiera wskaznik na typ calkowity, sluzzy do alokacji pamieci na dynamiczna tablice

Definicja w linii 21 pliku ListaTab.h.

5.4.4.2 unsigned int ListaTab::_RozmiarL [private]

Pole Klasy [ListaTab](#).

Pole przechowuje informacje o ilosci obecnie znajdujacych sie elementow na liscie danych

Definicja w linii 27 pliku ListaTab.h.

5.4.4.3 unsigned int ListaTab::_RozmiarT [private]

Pole Klasy [ListaTab](#).

Pole przechowuje informacje o obecnym rozmiarze tablicy danych

Definicja w linii 33 pliku ListaTab.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [ListaTab.h](#)
- [ListaTab.cpp](#)

5.5 Dokumentacja klasy Stos

Modeluje pojecie Stosu.

```
#include <Stos.h>
```

Diagram dziedziczenia dla Stos

Diagram współpracy dla Stos:

Komponenty

- struct [Wezel](#)

Metody publiczne

- void [_Push](#) (int k, unsigned int Pozycja=0)
Metoda dodajaca nowy [Wezel](#).
- int [_Pop](#) (unsigned int Pozycja=0)
Metda usuwajaca wezel.
- unsigned int [_Rozmiar](#) ()
Metoda informujaca o obecnej ilosci Wezlow.
- [Stos](#) ()
Konstruktor.
- [~Stos](#) ()
Destruktor Obiektu Destruktor przy pomocy funkcji pop usuwa wszystkie elementy ze stosu.
- [Stos](#) (const [Stos](#) &S)
Konstruktor Kopiujacy.
- void [_Pokaz](#) ()
Metoda wyswietlajaca elementy Stosu.
- void [_Zwolnij](#) ()
Metoda zwalnijaca pamiec zajeta przez struktre.

Atrybuty prywatne

- [Wezel](#) * [Gora](#)
Pole klasy [Stos](#) Pole jest wskaźnikiem na ostatnio dodany [Wezel](#).
- unsigned int [_Ilosc](#)
Pole klasy [Stos](#) Pole przechowuje informacje o ilosci wezlow.

5.5.1 Opis szczegółowy

Modeluje pojecie Stosu.

Klasa modeluje pojecie Kolejki, dodajac nowy element na jej koniec i sciagajac ostatnio dodany [Wezel](#)

Definicja w linii 20 pliku Stos.h.

5.5.2 Dokumentacja konstruktora i destruktor

5.5.2.1 Stos::Stos () [inline]

Konstruktor.

Konstruktor ma za zadanie ustawić wskaźnik na NULL oraz wyzerować ilość elementów podczas tworzenia obiektu tej klasy

Definicja w linii 82 pliku Stos.h.

5.5.2.2 Stos::~~Stos ()

Destruktor Obiektu Destruktor przy pomocy funkcji pop usuwa wszystkie elementy ze stosu.

Definicja w linii 23 pliku Stos.cpp.

5.5.2.3 Stos::Stos (const Stos & S)

Konstruktor Kopiujący.

5.5.3 Dokumentacja funkcji składowych

5.5.3.1 void Stos::_Pokaz () [virtual]

Metoda wyświetlająca elementy Stosu.

Metoda ma za zadanie wyświetlić wszystkie wartości znajdujące się na Stosie

Implementuje [Struktury](#).

Definicja w linii 55 pliku Stos.cpp.

5.5.3.2 int Stos::_Pop (unsigned int Pozycja = 0) [virtual]

Metoda usuwająca węzeł.

Metoda ma za zadanie zdjąć ostatnio dodany element ze stosu danych oraz zwrócić przechowywaną wartość

Zwraca

Zwraca ostatnią dodaną wartość

Implementuje [Struktury](#).

Definicja w linii 40 pliku Stos.cpp.

5.5.3.3 void Stos::_Push (int k, unsigned int Pozycja = 0) [virtual]

Metoda dodająca nowy [Węzeł](#).

Metoda ma za zadanie dodać nowy węzeł na koniec, umieścić w nim wartość zadana jako argument oraz pokazać wskaźnikiem na ostatni [Węzeł](#) przed dodaniem nowego Węzła

Parametry

<code>in</code>	<code>k</code>	- Wartość która zostanie umieszczona w odpowiednim polu Wezła
-----------------	----------------	---

Implementuje [Struktury](#).

Definicja w linii 31 pliku Stos.cpp.

5.5.3.4 `unsigned int Stos::_Rozmiar () [inline],[virtual]`

Metoda informująca o obecnej ilości Wezłów.

Metoda zwraca informacje o ilości obecnie znajdujących się elementów na stosie

Zwraca

- Zwraca ilość elementów na Stosie

Implementuje [Struktury](#).

Definicja w linii 75 pliku Stos.h.

5.5.3.5 `void Stos::_Zwolnij () [virtual]`

Metoda zwalniająca pamięć zajęta przez struktury.

Metoda ma za zadanie zwolnić pamięć zajęta przez załadowane do struktury dane, elementy są usuwane dopóki wskaźnik pokazujący na początek kolejki nie będzie wskazywał na NULL

Implementuje [Struktury](#).

Definicja w linii 7 pliku Stos.cpp.

5.5.4 Dokumentacja atrybutów składowych

5.5.4.1 `unsigned int Stos::_Ilosc [private]`

Pole klasy [Stos](#) Pole przechowuje informacje o ilości wezłów.

Definicja w linii 47 pliku Stos.h.

5.5.4.2 `Wezel* Stos::Gora [private]`

Pole klasy [Stos](#) Pole jest wskaźnikiem na ostatnio dodany [Wezel](#).

Definicja w linii 42 pliku Stos.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Stos.h](#)
- [Stos.cpp](#)

5.6 Dokumentacja klasy StosTab

```
#include <StosTab.h>
```

Diagram dziedziczenia dla StosTab

Diagram współpracy dla StosTab:

Metody publiczne

- [StosTab](#) ()
Konstruktor Podczas tworzenia obiektu tej klasy automatycznie alokowana jest tablica o rozmiarze 1 oraz ustawienie obecnej liczby elementow listy na 0.
- [~StosTab](#) ()
Konstruktor Kopiujacy.
- void [_Pokaz](#) ()
Metoda wypisujaca elementy Stosu.
- int [_Pop](#) (unsigned int Pozycja=0)
Metoda sciagajaca element ze stosu.
- void [_Push](#) (int k, unsigned int Pozycja=0)
Metoda dodajaca element do tablicy.
- unsigned int [_Rozmiar](#) ()
Metoda zwracajaca rozmiar listy.
- void [_Zwolnij](#) ()

Atrybuty prywatne

- int * [_L](#)
Pole klasy [StosTab](#).
- unsigned int [_RozmiarL](#)
Pole Klasy [StosTab](#).
- unsigned int [_RozmiarT](#)
Pole Klasy [StosTab](#).

5.6.1 Opis szczegółowy

Definicja w linii 13 pliku StosTab.h.

5.6.2 Dokumentacja konstruktora i destruktor

5.6.2.1 [StosTab::StosTab](#) ()

Konstruktor Podczas tworzenia obiektu tej klasy automatycznie alokowana jest tablica o rozmiarze 1 oraz ustawienie obecnej liczby elementow listy na 0.

Definicja w linii 59 pliku StosTab.cpp.

5.6.2.2 [StosTab::~~StosTab](#) () [inline]

Konstruktor Kopiujacy.

Destruktor obiektu

Definicja w linii 48 pliku StosTab.h.

5.6.3 Dokumentacja funkcji składowych

5.6.3.1 void [StosTab::_Pokaz](#) () [virtual]

Metoda wypisujaca elementy Stosu.

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

Implementuje [Struktury](#).

Definicja w linii 68 pliku StosTab.cpp.

5.6.3.2 int StosTab::_Pop (unsigned int *Pozycja* = 0) [virtual]

Metoda ściągająca element ze stosu.

Metoda ma za zadanie ściągnąć ostatni element stosu, w przypadku gdy tablica jest do połowy pusta następuje utworzenie nowej tablicy o dwa razy mniejszym rozmiarze

Parametry

in	<i>Pozycja</i>	- numer elementy który zostanie usunięty z listy i zostanie zwrócona jego wartość
----	----------------	---

Zwraca

Zwraca wybrany przez użytkownika element

Implementuje [Struktury](#).

Definicja w linii 13 pliku StosTab.cpp.

5.6.3.3 void StosTab::_Push (int *k*, unsigned int *Pozycja* = 0) [virtual]

Metoda dodająca element do tablicy.

Metoda ma za zadanie dodać nowy element na końcu stosu, w przypadku zapelnienia tablicy następuje utworzenie nowej tablicy i przepisanie elementów

Parametry

in	<i>k</i>	- wartość jaką chcemy dodać do listy
in	<i>Pozycja</i>	- Pozycja na której chcemy dodać wartość

Implementuje [Struktury](#).

Definicja w linii 39 pliku StosTab.cpp.

5.6.3.4 unsigned int StosTab::_Rozmiar () [inline],[virtual]

Metoda zwracająca rozmiar listy.

Metoda zwraca informacje o obecnej ilości danych w strukturze

Zwraca

Zwraca ilość elementów listy

Implementuje [Struktury](#).

Definicja w linii 86 pliku StosTab.h.

5.6.3.5 void StosTab::_Zwolnij () [virtual]

Metoda zwalniania pamięci

Metoda ma za zadanie zwolnić pamięć zajmowaną przez dane, dopóki ilość elementów listy nie wynosi 0 wykonywana jest metoda `_Pop`, aby opróżnić stos i zwolnić pamięć

Implementuje [Struktury](#).

Definicja w linii 7 pliku StosTab.cpp.

5.6.4 Dokumentacja atrybutów składowych

5.6.4.1 `int* StosTab::_L` `[private]`

Pole klasy [StosTab](#).

Pole zawiera wskaźnik na typ całkowity, służy do alokacji pamięci na dynamiczną tablicę

Definicja w linii 21 pliku StosTab.h.

5.6.4.2 `unsigned int StosTab::_RozmiarL` `[private]`

Pole Klasy [StosTab](#).

Pole przechowuje informacje o ilości obecnie znajdujących się elementów na liście danych

Definicja w linii 27 pliku StosTab.h.

5.6.4.3 `unsigned int StosTab::_RozmiarT` `[private]`

Pole Klasy [StosTab](#).

Pole przechowuje informacje o obecnym rozmiarze tablicy danych

Definicja w linii 33 pliku StosTab.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [StosTab.h](#)
- [StosTab.cpp](#)

5.7 Dokumentacja klasy Struktury

Modeluje pojęcie [Struktury](#) danych, klasa bazowa dla Stosu, Kolejki i Listy, zarówno w implementacji wskaźnikowej jak i tablicowej.

```
#include <Struktury.h>
```

Diagram dziedziczenia dla Struktury

Metody publiczne

- `virtual void _Push (int k, unsigned int Pozycja=1)=0`
Metoda dodająca kolejny element struktury.
- `virtual int _Pop (unsigned int Pozycja=1)=0`
Metoda usuwająca element.
- `virtual unsigned int _Rozmiar ()=0`
Metoda zwracająca rozmiar [Struktury](#).
- `virtual void _Pokaz ()=0`
Metoda wyświetlająca dane.
- `virtual void _Zwolnij ()=0`
Metoda zwalniania pamięć.

5.7.1 Opis szczegółowy

Modeluje pojecie [Struktury](#) danych, klasa bazowa dla Stosu, Kolejki i Listy, zarowno w implemenetacji wskaznikowej jak i tablicowej.

Definicja w linii 20 pliku Struktury.h.

5.7.2 Dokumentacja funkcji składowych

5.7.2.1 `virtual void Struktury::_Pokaz () [pure virtual]`

Metoda wyswietlajaca dane.

Metoda ma za zadanie wyswietlic wszystkie dane nalezace do struktury

Implementowany w [Stos](#), [TabListaPod](#), [Lista](#), [Kolejka](#), [ListaTab](#) i [StosTab](#).

5.7.2.2 `virtual int Struktury::_Pop (unsigned int Pozycja = 1) [pure virtual]`

Metoda usuwajaca element.

Metoda ma za zadanie usunac element i w zaleznosci od implementowanej struktury bedzie to usuwany element usuwany z poczatku, konca lub w przypadku listy z dowolnego jej miejsca

Parametry

<code>in</code>	<i>Pozycja</i>	- Numer elementu ,ktory zostanie dodany. Argument ma znaczenie tylko w przypadku listy i domyślnie jest ustawiony, tak aby element był dodawany zawsze na poczatku listy
-----------------	----------------	--

Zwraca

Zwraca wartosc elementu z odpowiedniego dla wybranej struktury miejsca

Implementowany w [Kolejka](#), [ListaTab](#), [Stos](#), [StosTab](#), [TabListaPod](#) i [Lista](#).

5.7.2.3 `virtual void Struktury::_Push (int k, unsigned int Pozycja = 1) [pure virtual]`

Metoda dodajaca kolejny element struktury.

Metoda ma za zadanie dodac kolejny element do naszej struktury oraz zapisac w nim odpowiednia wartosc. W zaleznosci od implementowanej struktury element bedzie dodawany na poczatku lub na koncu struktury danych.

Parametry

<code>in</code>	<i>k</i>	- wartosc typu całkowitego, ktora bedzie umieszczona w strukturze
-----------------	----------	---

Implementowany w [Kolejka](#), [ListaTab](#), [StosTab](#), [TabListaPod](#), [Stos](#) i [Lista](#).

5.7.2.4 `virtual unsigned int Struktury::_Rozmiar () [pure virtual]`

Metoda zwracajaca rozmiar [Struktury](#).

Metoda ma zadanie zwrocic bierzaca liczbe elementow nalezacych do danej struktury

Zwraca

- Bierzaca liczba elementow [Struktury](#) danych

Implementowany w [Kolejka](#), [ListaTab](#), [StosTab](#), [TabListaPod](#), [Stos](#) i [Lista](#).

5.7.2.5 virtual void Struktury::_Zwolnij() [pure virtual]

Metoda zwalnijająca pamięć.

Metoda ma za zadanie zwolnić pamięć używaną przy zapelnianiu danej struktury danymi

Implementowany w [Kolejka](#), [Stos](#), [TabListaPod](#), [Lista](#), [ListaTab](#) i [StosTab](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Struktury.h](#)

5.8 Dokumentacja klasy StrukturyBenchmark

```
#include <StrukturyBenchmark.h>
```

Diagram dziedziczenia dla StrukturyBenchmark

Diagram współpracy dla StrukturyBenchmark:

Metody publiczne

- void [_Test](#) (const unsigned int n)
Metoda wykonująca test dla odpowiedniej struktury.
- void [_Wczytaj](#) (string PlikIn, const unsigned int Ilosc)
Metoda Wczytująca dane.
- void [_Ustaw](#) ([Struktury](#) &K)
Metoda przypisująca obiekt danej struktury.
- [StrukturyBenchmark](#) ()
Konstruktor.
- void [_Przydziel](#) (const unsigned int n)
Metoda alokująca pamięć.
- virtual [~StrukturyBenchmark](#) ()
Destruktor.
- void [_Zwolnij](#) (const unsigned int n)
Metoda zwalnijająca zaalokowana przez struktury pamięć.

Atrybuty prywatne

- [Struktury](#) * [S](#)
Pole StrukturyBenchmark Pole zawiera wskaźnik na [Struktury](#), za pomocą niego i metod wirtualnych będą wywoływane odpowiednie dla danej struktury metody.
- int * [W](#)
Pole StrukturyBenchmark Pole zawiera wskaźnik na typ całkowity, służy on do alokowania pamięci dla wczytanych z pliku danych.

5.8.1 Opis szczegółowy

Klasa modeluje pojęcie Benchmarku przeznaczonego dla struktur danych przechowujące dane

Definicja w linii 18 pliku StrukturyBenchmark.h.

5.8.2 Dokumentacja konstruktora i destruktora

5.8.2.1 StrukturyBenchmark::StrukturyBenchmark () [inline]

Konstruktor.

Konstruktor sprawia ze wskazniki nowo powstałego obiektu wskazują na NULL

Definicja w linii 66 pliku StrukturyBenchmark.h.

5.8.2.2 virtual StrukturyBenchmark::~~StrukturyBenchmark () [inline],[virtual]

Destruktor.

Definicja w linii 79 pliku StrukturyBenchmark.h.

5.8.3 Dokumentacja funkcji składowych

5.8.3.1 void StrukturyBenchmark::_Przydziel (const unsigned int *n*)

Metoda alokująca pamięć.

Metoda ma za zadanie zaalokować odpowiednią ilość danych w zależności od tego ile zostało ich wczytanych

Parametry

<i>in</i>	<i>n</i>	- Ilość wczytanych danych
-----------	----------	---------------------------

Definicja w linii 13 pliku StrukturyBenchmark.cpp.

Oto graf wywołań tej funkcji:

5.8.3.2 void StrukturyBenchmark::_Test (const unsigned int *n*) [virtual]

Metoda wykonująca test dla odpowiedniej struktury.

Metoda ma za zadanie wykonać zapełnienie struktury danymi o zadanej w argumencie ilości

Parametry

<i>in</i>	<i>n</i>	- ilość danych która zapełniła struktura
-----------	----------	--

Implementuje [BenchmarkInterfejs](#).

Definicja w linii 20 pliku StrukturyBenchmark.cpp.

5.8.3.3 void StrukturyBenchmark::_Ustaw (Struktury & *K*) [inline]

Metoda przypisująca obiekt danej struktury.

Metoda ma za zadanie pokazywać na testowaną obecnie strukturę danych, aby za pomocą wskaźnika wywoływać metody z odpowiednich struktur

Parametry

<i>in</i>	<i>K</i>	- Adres aktualnie testowanego obiektu danej struktury
-----------	----------	---

Definicja w linii 60 pliku StrukturyBenchmark.h.

Oto graf wywołań tej funkcji:

5.8.3.4 void StrukturyBenchmark::_Wczytaj (string *PlikIn*, const unsigned int *Ilosc*)

Metoda Wczytujaca dane.

Metoda ma za zadanie wczytac dane wejciowe o podanej przez argument nazwie oraz przypisac wskaznik do zaalokowanych w pamieci danych

Parametry

in	<i>PlikIn</i>	- nazwa pliku wejsciowego z danymi
in	<i>Ilosc</i>	- Ilosc danych jaka bedzie wczytywana

Definicja w linii 27 pliku StrukturyBenchmark.cpp.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

5.8.3.5 void StrukturyBenchmark::_Zwolnij (const unsigned int *n*) [virtual]

Metoda zwalnijaca zaalokowana przez struktury pamiec.

Metoda ma za zadanie oproznic zaladowane do struktury dane

Parametry

in	<i>n</i>	- Ilosc danych ktora zostanie zwolniona
----	----------	---

Implementuje [BenchmarkInterfejs](#).

5.8.4 Dokumentacja atrybutów składowych

5.8.4.1 Struktury* StrukturyBenchmark::S [private]

Pole StrukturyBenchmark Pole zawiera wskaznik na [Struktury](#), za pomoca niego i metod wirtualnych beda wywolowane odpowiednie dla danej struktury metody.

Definicja w linii 26 pliku StrukturyBenchmark.h.

5.8.4.2 int* StrukturyBenchmark::W [private]

Pole StrukturyBenchmark Pole zawiera wskaznik na typ calkowity, sluzy on do alokowania pamieci dla wczytanych z pliku danych.

Definicja w linii 33 pliku StrukturyBenchmark.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [StrukturyBenchmark.h](#)
- [StrukturyBenchmark.cpp](#)

5.9 Dokumentacja klasy StrukturyTablice

```
#include <StrukturyTablice.h>
```

Metody publiczne

- virtual void [_Push](#) (int k, unsigned int Pozycja=0)=0
- virtual int [_Pop](#) (unsigned int Pozycja=0)=0

- virtual void [_Pokaz](#) ()=0
- virtual unsigned int [_Rozmiar](#) ()=0

5.9.1 Opis szczegółowy

Definicja w linii 8 pliku StrukturyTablice.h.

5.9.2 Dokumentacja funkcji składowych

5.9.2.1 virtual void StrukturyTablice::_Pokaz () [pure virtual]

5.9.2.2 virtual int StrukturyTablice::_Pop (unsigned int *Pozycja* = 0) [pure virtual]

5.9.2.3 virtual void StrukturyTablice::_Push (int *k*, unsigned int *Pozycja* = 0) [pure virtual]

5.9.2.4 virtual unsigned int StrukturyTablice::_Rozmiar () [pure virtual]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [StrukturyTablice.h](#)

5.10 Dokumentacja klasy TabListaPod

```
#include <TabListaPod.h>
```

Diagram dziedziczenia dla TabListaPod

Diagram współpracy dla TabListaPod:

Metody publiczne

- [TabListaPod](#) (const [TabListaPod](#) &K)
Konstruktor Kopiujacy.
- [TabListaPod](#) ()
Konstruktor Podczas tworzenia obiektu tej klasy automatycznie alokowana jest tablica o rozmiarze 1 oraz ustawienie obecnej liczby elementow listy na 0.
- [~TabListaPod](#) ()
Destruktor obiektu.
- int [_Pop](#) (unsigned int *Pozycja*)
Metoda scigajaca element z listy Metoda sciagnac element i w przypadku ,gdy rozmiar tablicy bedzie dwukrotnie wiekszy od ilosci elementow, zostanie zalokowana nowa tablica o zmniejszonym dwukrotnie rozmiarze.
- void [_Push](#) (int *k*, unsigned int *Pozycja*)
Dodaje element do listy.
- unsigned int [_Rozmiar](#) ()
Metoda zwracajaca rozmiar Listy.
- void [_Pokaz](#) ()
Metoda wypisujaca elementy listy.
- void [_Zwolnij](#) ()
- void [_XRozszerz](#) (const double *l1*, unsigned int *Pozycja*, int *k*=0)

Atrybuty prywatne

- `int * _L`
Pole klasy [ListaTab](#).
- `unsigned int _RozmiarL`
Pole Klasy [ListaTab](#).
- `unsigned int _RozmiarT`
Pole Klasy [ListaTab](#).

5.10.1 Opis szczegółowy

Definicja w linii 13 pliku `TabListaPod.h`.

5.10.2 Dokumentacja konstruktora i destruktora

5.10.2.1 `TabListaPod::TabListaPod (const TabListaPod & K)`

Konstruktor Kopiujacy.

Definicja w linii 8 pliku `TabListaPod.cpp`.

5.10.2.2 `TabListaPod::TabListaPod ()`

Konstruktor Podczas tworzenia obiektu tej klasy automatycznie alokowana jest tablica o rozmiarze 1 oraz ustawienie obecnej liczby elementow listy na 0.

Definicja w linii 41 pliku `TabListaPod.cpp`.

5.10.2.3 `TabListaPod::~~TabListaPod () [inline]`

Destruktor obiektu.

Definicja w linii 49 pliku `TabListaPod.h`.

5.10.3 Dokumentacja funkcji składowych

5.10.3.1 `void TabListaPod::_Pokaz () [virtual]`

Metoda wypisujaca elementy listy.

Metoda ma za zadanie wypisac wszystkie elementy znajdujace sie obecnie na liscie danych

Implementuje [Struktury](#).

Definicja w linii 48 pliku `TabListaPod.cpp`.

5.10.3.2 `int TabListaPod::_Pop (unsigned int Pozycja) [virtual]`

Metoda scigajaca element z listy Metoda sciagnac element i w przypadku ,gdy rozmiar tablicy bedzie dwukrotnie wiekszy od ilosci elementow, zostanie zalokowana nowa tablica o zmniejszonym dwukrotnie rozmiarze.

Parametry

in	Pozycja	- numer elementu który zostanie zwrócony z listy
----	---------	--

Zwraca

Zwraca wartosc elementu z zadanej pozycji

Implementuje [Struktury](#).

Definicja w linii 59 pliku TabListaPod.cpp.

5.10.3.3 void TabListaPod::_Push (int k, unsigned int Pozycja) [virtual]

Dodaje element do listy.

Metoda ma za zadanie dodac element do listy. W przypadku, gdy skonczy sie miejsce w tablicy, zostaje wówczas alokowana nowa tablica o dwukrotnie wiekszym rozmiarze, w inny przypadku element zostaje dodany w miejscu zadany przez uzytkownika poprzez argument metody

Parametry

in	k	- Wartosc jaka chcemy umiescic na liscie
in	Pozycja	- Pozycja na ktorej chcemy dodac element

Implementuje [Struktury](#).

Definicja w linii 21 pliku TabListaPod.cpp.

5.10.3.4 unsigned int TabListaPod::_Rozmiar () [inline], [virtual]

Metoda zwracajaca rozmiar Listy.

Metoda ma za zadanie zwrocic informacje o aktualnej ilosci elementow na liscie

Zwraca

Liczba elementow listy

Implementuje [Struktury](#).

Definicja w linii 84 pliku TabListaPod.h.

5.10.3.5 void TabListaPod::_XRozszerz (const double Ile, unsigned int Pozycja, int k = 0)

Definicja w linii 85 pliku TabListaPod.cpp.

5.10.3.6 void TabListaPod::_Zwolnij () [virtual]

Metoda zwalnijaca pamiec

Metoda ma za zadanie zwolnij pamiec zajmowana przez dane, dopoki ilosc elementow listy nie wynosi 0 wykonywana jest metoda _Pop, aby oproznic stos i zwolnic pamiec

Implementuje [Struktury](#).

Definicja w linii 15 pliku TabListaPod.cpp.

5.10.4 Dokumentacja atrybutów składowych

5.10.4.1 `int* TabListaPod::_L` `[private]`

Pole klasy [ListaTab](#).

Pole zawiera wskaźnik na typ całkowity, służy do alokacji pamięci na dynamiczną tablicę

Definicja w linii 21 pliku `TabListaPod.h`.

5.10.4.2 `unsigned int TabListaPod::_RozmiarL` `[private]`

Pole klasy [ListaTab](#).

Pole przechowuje informacje o ilości obecnie znajdujących się elementów na liście danych

Definicja w linii 27 pliku `TabListaPod.h`.

5.10.4.3 `unsigned int TabListaPod::_RozmiarT` `[private]`

Pole klasy [ListaTab](#).

Pole przechowuje informacje o obecnym rozmiarze tablicy danych

Definicja w linii 33 pliku `TabListaPod.h`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [TabListaPod.h](#)
- [TabListaPod.cpp](#)

5.11 Dokumentacja struktury `Stos::Wezel`

Diagram współpracy dla `Stos::Wezel`:

Atrybuty publiczne

- `Wezel * _Nast`
Pole Wezła Pole będące wskaźnikiem na kolejny element stosu.
- `int _Wartosc`
Pole Wezła Pole przechowuje wartość typu całkowitego.

5.11.1 Opis szczegółowy

Klasy [Stos](#) Pole modeluje pojęcie węzła, będącego podstawą dla implementacji struktury danych w rozwinięciu wskaźnikowym

Definicja w linii 27 pliku `Stos.h`.

5.11.2 Dokumentacja atrybutów składowych

5.11.2.1 `Wezel* Stos::Wezel::_Nast`

Pole Wezła Pole będące wskaźnikiem na kolejny element stosu.

Definicja w linii 32 pliku `Stos.h`.

5.11.2.2 int Stos::Wezel::_Wartosc

Pole Wezla Pole przechowuje wartosc typu calkowitego.

Definicja w linii 36 pliku Stos.h.

Dokumentacja dla tej struktury zostala wygenerowana z pliku:

- [Stos.h](#)

5.12 Dokumentacja struktury Lista::Wezel

Diagram wspolpracy dla Lista::Wezel:

Atrybuty publiczne

- [Wezel](#) * [_Nast](#)

Pole Wezla Pole bedace wskaznikiem na kolejny element listy.

- int [_Wartosc](#)

Pole Wezla Pole przechowuje wartosc typu calkowitego.

5.12.1 Opis szczegolowy

Klasy lista Pole modelje pojecie wezla,bedacego podstawa dla implementacji struktury danych w rozwiniECiu wskaznikowym

Definicja w linii 19 pliku Lista.h.

5.12.2 Dokumentacja atrybutow skadowych

5.12.2.1 Wezel* Lista::Wezel::_Nast

Pole Wezla Pole bedace wskaznikiem na kolejny element listy.

Definicja w linii 24 pliku Lista.h.

5.12.2.2 int Lista::Wezel::_Wartosc

Pole Wezla Pole przechowuje wartosc typu calkowitego.

Definicja w linii 28 pliku Lista.h.

Dokumentacja dla tej struktury zostala wygenerowana z pliku:

- [Lista.h](#)

5.13 Dokumentacja struktury Kolejka::Wezel

Diagram wspolpracy dla Kolejka::Wezel:

Atrybuty publiczne

- [Wezel](#) * [_Nast](#)
Pole Wezla Pole bedace wskaznikiem na kolejny element Kolejki.
- [int](#) [_Wartosc](#)
Pole Wezla Pole przechowuje wartosc typu calkowitego.

5.13.1 Opis szczegółowy

Klasy [Kolejka](#) Pole modelje pojecie wezla,bedacego podstawa dla implementacji struktury danych w rozwinięciu wskaznikowym

Definicja w linii 29 pliku Kolejka.h.

5.13.2 Dokumentacja atrybutów składowych

5.13.2.1 [Wezel](#)* [Kolejka::Wezel::_Nast](#)

Pole Wezla Pole bedace wskaznikiem na kolejny element Kolejki.

Definicja w linii 34 pliku Kolejka.h.

5.13.2.2 [int](#) [Kolejka::Wezel::_Wartosc](#)

Pole Wezla Pole przechowuje wartosc typu calkowitego.

Definicja w linii 38 pliku Kolejka.h.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Kolejka.h](#)

Rozdział 6

Dokumentacja plików

6.1 Dokumentacja pliku BenchmarkInterfejs.cpp

Definicje Metod klasy [BenchmarkInterfejs](#).

```
#include "BenchmarkInterfejs.h"
#include <sys/time.h>
#include <iostream>
#include <fstream>
```

Wykres zależności załączania dla BenchmarkInterfejs.cpp:

6.1.1 Opis szczegółowy

Definicje Metod klasy [BenchmarkInterfejs](#).

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

6.2 Dokumentacja pliku BenchmarkInterfejs.h

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <cstring>
#include "Struktury.h"
```

Wykres zależności załączania dla BenchmarkInterfejs.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [BenchmarkInterfejs](#)

Modeluje pojecie Interfejsu Benchmark'u.

6.3 Dokumentacja pliku Kolejka.cpp

Definicje Metod klasy [Kolejka](#).

```
#include "Kolejka.h"
```

Wykres zależności załączania dla Kolejka.cpp:

6.3.1 Opis szczegółowy

Definicje Metod klasy [Kolejka](#).

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

6.4 Dokumentacja pliku Kolejka.h

```
#include <iostream>
#include <cstdlib>
#include "Struktury.h"
```

Wykres zależności załączania dla Kolejka.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [Kolejka](#)
Modeluje pojecie Kolejki.
- struct [Kolejka::Wezel](#)

6.5 Dokumentacja pliku Lista.cpp

Definicje Metod klasy [Lista](#).

```
#include "Lista.h"
```

Wykres zależności załączania dla Lista.cpp:

6.5.1 Opis szczegółowy

Definicje Metod klasy [Lista](#).

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

6.6 Dokumentacja pliku Lista.h

```
#include <iostream>
#include "Struktury.h"
```

Wykres zależności załączania dla Lista.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [Lista](#)
- struct [Lista::Wezel](#)

6.7 Dokumentacja pliku ListaTab.cpp

Zawiera definicje Metod klasy [ListaTab](#).

```
#include "ListaTab.h"
```

Wykres zależności załączania dla ListaTab.cpp:

6.7.1 Opis szczegółowy

Zawiera definicje Metod klasy [ListaTab](#).

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

6.8 Dokumentacja pliku ListaTab.h

```
#include <cstdlib>
#include <iostream>
#include "Struktury.h"
```

Wykres zależności załączania dla ListaTab.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [ListaTab](#)

6.9 Dokumentacja pliku Main.cpp

funkcja glowna programu

```
#include "ListaTab.h"
#include "TabListaPod.h"
#include "Lista.h"
#include "Stos.h"
#include "Kolejka.h"
#include "StosTab.h"
#include "StrukturyBenchmark.h"
```

Wykres zależności załączania dla Main.cpp:

Definicje

- #define [ILOSC_DANYCH](#) 100000000

Funkcje

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

6.9.1 Opis szczegółowy

funkcja glowna programu

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

6.9.2 Dokumentacja definicji

6.9.2.1 #define ILOSC_DANYCH 100000000

Definicja w linii 8 pliku Main.cpp.

6.9.3 Dokumentacja funkcji

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

Definicja w linii 14 pliku `Main.cpp`.

Oto graf wywołań dla tej funkcji:

6.10 Dokumentacja pliku `Stos.cpp`

Definicje metod klasy [Stos](#).

```
#include "Stos.h"
```

Wykres zależności załączania dla `Stos.cpp`:

6.10.1 Opis szczegółowy

Definicje metod klasy [Stos](#).

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

6.11 Dokumentacja pliku `Stos.h`

```
#include <iostream>
```

```
#include "Struktury.h"
```

Wykres zależności załączania dla `Stos.h`: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [Stos](#)
Modeluje pojecie Stosu.
- struct [Stos::Wezel](#)

6.12 Dokumentacja pliku `StosTab.cpp`

Zawiera definicje Metod klasy [ListaTab](#).

```
#include "StosTab.h"
```

Wykres zależności załączania dla `StosTab.cpp`:

6.12.1 Opis szczegółowy

Zawiera definicje Metod klasy [ListaTab](#).

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

6.13 Dokumentacja pliku `StosTab.h`

```
#include <cstdlib>
```

```
#include <iostream>
#include "Struktury.h"
```

Wykres zależności załączania dla StosTab.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [StosTab](#)

6.14 Dokumentacja pliku strona-glowna.dox

6.15 Dokumentacja pliku Struktury.h

```
#include <iostream>
#include <cstring>
#include <sstream>
#include <fstream>
```

Wykres zależności załączania dla Struktury.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [Struktury](#)

Modeluje pojęcie [Struktury](#) danych, klasa bazowa dla Stosu, Kolejki i Listy, zarówno w implementacji wskaźnikowej jak i tablicowej.

6.16 Dokumentacja pliku StrukturyBenchmark.cpp

Definicje metod klasy [StrukturyBenchmark](#).

```
#include "StrukturyBenchmark.h"
```

Wykres zależności załączania dla StrukturyBenchmark.cpp:

6.16.1 Opis szczegółowy

Definicje metod klasy [StrukturyBenchmark](#).

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

6.17 Dokumentacja pliku StrukturyBenchmark.h

```
#include "BenchmarkInterfejs.h"
#include "Struktury.h"
```

Wykres zależności załączania dla StrukturyBenchmark.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [StrukturyBenchmark](#)

6.18 Dokumentacja pliku StrukturyTablice.h

```
#include <iostream>
#include <cstring>
#include <sstream>
#include <fstream>
```

Wykres zależności załączania dla StrukturyTablice.h:

Komponenty

- class [StrukturyTablice](#)

6.19 Dokumentacja pliku TabListaPod.cpp

Definicje Metod Klasy [TabListaPod](#).

```
#include "TabListaPod.h"
```

Wykres zależności załączania dla TabListaPod.cpp:

6.19.1 Opis szczegółowy

Definicje Metod Klasy [TabListaPod](#).

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

6.20 Dokumentacja pliku TabListaPod.h

```
#include <cstdlib>
#include <iostream>
#include "Struktury.h"
```

Wykres zależności załączania dla TabListaPod.h: Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [TabListaPod](#)

Skorowidz

- ~Kolejka
 - Kolejka, [12](#)
- ~Lista
 - Lista, [14](#)
- ~ListaTab
 - ListaTab, [17](#)
- ~Stos
 - Stos, [20](#)
- ~StosTab
 - StosTab, [22](#)
- ~StrukturyBenchmark
 - StrukturyBenchmark, [27](#)
- ~TabListaPod
 - TabListaPod, [30](#)
- _Generator
 - BenchmarkInterfejs, [9](#)
- _Ilosc
 - Kolejka, [13](#)
 - Lista, [15](#)
 - Stos, [21](#)
- _L
 - ListaTab, [18](#)
 - StosTab, [24](#)
 - TabListaPod, [32](#)
- _Nast
 - Kolejka::Wezel, [34](#)
 - Lista::Wezel, [33](#)
 - Stos::Wezel, [32](#)
- _Ostatni
 - Kolejka, [13](#)
- _Pierwszy
 - Kolejka, [13](#)
- _Pokaz
 - Kolejka, [12](#)
 - Lista, [14](#)
 - ListaTab, [17](#)
 - Stos, [20](#)
 - StosTab, [22](#)
 - Struktury, [25](#)
 - StrukturyTablice, [29](#)
 - TabListaPod, [30](#)
- _Pop
 - Kolejka, [12](#)
 - Lista, [15](#)
 - ListaTab, [17](#)
 - Stos, [20](#)
 - StosTab, [23](#)
 - Struktury, [25](#)
 - StrukturyTablice, [29](#)
- TabListaPod, [30](#)
- _Przydziel
 - StrukturyBenchmark, [27](#)
- _Push
 - Kolejka, [12](#)
 - Lista, [15](#)
 - ListaTab, [17](#)
 - Stos, [20](#)
 - StosTab, [23](#)
 - Struktury, [25](#)
 - StrukturyTablice, [29](#)
 - TabListaPod, [31](#)
- _Rozmiar
 - Kolejka, [12](#)
 - Lista, [15](#)
 - ListaTab, [18](#)
 - Stos, [21](#)
 - StosTab, [23](#)
 - Struktury, [25](#)
 - StrukturyTablice, [29](#)
 - TabListaPod, [31](#)
- _RozmiarL
 - ListaTab, [18](#)
 - StosTab, [24](#)
 - TabListaPod, [32](#)
- _RozmiarT
 - ListaTab, [18](#)
 - StosTab, [24](#)
 - TabListaPod, [32](#)
- _Test
 - BenchmarkInterfejs, [10](#)
 - StrukturyBenchmark, [27](#)
- _Ustaw
 - StrukturyBenchmark, [27](#)
- _Wartosc
 - Kolejka::Wezel, [34](#)
 - Lista::Wezel, [33](#)
 - Stos::Wezel, [32](#)
- _Wczytaj
 - BenchmarkInterfejs, [10](#)
 - StrukturyBenchmark, [27](#)
- _WykonajTest
 - BenchmarkInterfejs, [10](#)
- _XRozszerz
 - TabListaPod, [31](#)
- _Zwolnij
 - BenchmarkInterfejs, [10](#)
 - Kolejka, [13](#)
 - Lista, [15](#)

ListaTab, 18
 Stos, 21
 StosTab, 23
 Struktury, 25
 StrukturyBenchmark, 28
 TabListaPod, 31

 BenchmarkInterfejs, 9
 _Generator, 9
 _Test, 10
 _Wczytaj, 10
 _WykonajTest, 10
 _Zwolnij, 10
 BenchmarkInterfejs.cpp, 35
 BenchmarkInterfejs.h, 35

 Glowa
 Lista, 16
 Gora
 Stos, 21

 ILOSC_DANYCH
 Main.cpp, 37

 Kolejka, 11
 ~Kolejka, 12
 _Ilosc, 13
 _Ostatni, 13
 _Pierwszy, 13
 _Pokaz, 12
 _Pop, 12
 _Push, 12
 _Rozmiar, 12
 _Zwolnij, 13
 Kolejka, 12
 Kolejka.cpp, 35
 Kolejka.h, 36
 Kolejka::Wezel, 33
 _Nast, 34
 _Wartosc, 34

 Lista, 13
 ~Lista, 14
 _Ilosc, 15
 _Pokaz, 14
 _Pop, 15
 _Push, 15
 _Rozmiar, 15
 _Zwolnij, 15
 Glowa, 16
 Lista, 14
 Lista.cpp, 36
 Lista.h, 36
 Lista::Wezel, 33
 _Nast, 33
 _Wartosc, 33
 ListaTab, 16
 ~ListaTab, 17
 _L, 18
 _Pokaz, 17
 _Pop, 17
 _Push, 17
 _Rozmiar, 18
 _RozmiarL, 18
 _RozmiarT, 18
 _Zwolnij, 18
 ListaTab, 17
 ListaTab, 17
 ListaTab.cpp, 36
 ListaTab.h, 37

 main
 Main.cpp, 38
 Main.cpp, 37
 ILOSC_DANYCH, 37
 main, 38

 S
 StrukturyBenchmark, 28
 Stos, 19
 ~Stos, 20
 _Ilosc, 21
 _Pokaz, 20
 _Pop, 20
 _Push, 20
 _Rozmiar, 21
 _Zwolnij, 21
 Gora, 21
 Stos, 20
 Stos.cpp, 38
 Stos.h, 38
 Stos::Wezel, 32
 _Nast, 32
 _Wartosc, 32
 StosTab, 21
 ~StosTab, 22
 _L, 24
 _Pokaz, 22
 _Pop, 23
 _Push, 23
 _Rozmiar, 23
 _RozmiarL, 24
 _RozmiarT, 24
 _Zwolnij, 23
 StosTab, 22
 StosTab, 22
 StosTab.cpp, 38
 StosTab.h, 38
 strona-glowna.dox, 39
 Struktury, 24
 _Pokaz, 25
 _Pop, 25
 _Push, 25
 _Rozmiar, 25
 _Zwolnij, 25
 Struktury.h, 39
 StrukturyBenchmark, 26
 ~StrukturyBenchmark, 27

- [_Przydziel, 27](#)
 - [_Test, 27](#)
 - [_Ustaw, 27](#)
 - [_Wczytaj, 27](#)
 - [_Zwolnij, 28](#)
 - [S, 28](#)
 - [StrukturyBenchmark, 27](#)
 - [StrukturyBenchmark, 27](#)
 - [W, 28](#)
- [StrukturyBenchmark.cpp, 39](#)
- [StrukturyBenchmark.h, 39](#)
- [StrukturyTablice, 28](#)
 - [_Pokaz, 29](#)
 - [_Pop, 29](#)
 - [_Push, 29](#)
 - [_Rozmiar, 29](#)
- [StrukturyTablice.h, 40](#)
- [TabListaPod, 29](#)
 - [~TabListaPod, 30](#)
 - [_L, 32](#)
 - [_Pokaz, 30](#)
 - [_Pop, 30](#)
 - [_Push, 31](#)
 - [_Rozmiar, 31](#)
 - [_RozmiarL, 32](#)
 - [_RozmiarT, 32](#)
 - [_XRozszerz, 31](#)
 - [_Zwolnij, 31](#)
 - [TabListaPod, 30](#)
 - [TabListaPod, 30](#)
- [TabListaPod.cpp, 40](#)
- [TabListaPod.h, 40](#)
- W
 - [StrukturyBenchmark, 28](#)