# PAMSI 3

Wygenerowano przez Doxygen 1.8.6

Pn, 17 mar 2014 01:23:48

# Spis treści

# 1 Strona główna

Laboratorium 3.

Rozne implementacje stosu i kolejki. Pomiar zlozonosci obliczeniowek wstawiania elementow do stosu zaimplentowanego na tablicy.

**Autor**

Jakub Chmiel 200314

# 2 Indeks klas

## 2.1 Lista klas

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



# 3 Indeks plików

## 3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

# 4 Dokumentacja klas

## 4.1 Dokumentacja struktury ElementKolejki

Struktura pojedynczego elementu przechowywanego w kolejce.

`#include <KolejkaLista.h>`

### Metody publiczne

- ElementKolejki ()

  *Konstruktor.*
- ∼ElementKolejki ()

  *Destruktor.*

### Atrybuty publiczne

- ElementKolejki ∗ nastepny

  *Wskaznik do nastepnego elementu.*
- ElementKolejki ∗ poprzedni

  *Wskaznik do poprzedniego elementu.*
- TYP elem

  *Wlasciwy zapamietany element.*

### 4.1.1 Opis szczegółowy

Struktura pojedynczego elementu przechowywanego w kolejce.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- KolejkaLista.h

## 4.2 Dokumentacja struktury ElementStosu

Struktura pojedynczego elementu przechowywanego w stosie.

`#include <StosLista.h>`

### Metody publiczne

- ElementStosu ()
  *Konstruktor.*
- ~ElementStosu ()
  *Destruktor.*

### Atrybuty publiczne

- ElementStosu * nastepny
  *Wskaznik do nastepnego elementu.*
- TYP elem
  *Wlasciwy zapamietany element.*

### 4.2.1 Opis szczegółowy

Struktura pojedynczego elementu przechowywanego w stosie.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- StosLista.h

## 4.3 Dokumentacja klasy KolejkaLista

Implementacja kolejki na liscie.

`#include <KolejkaLista.h>`

### Metody publiczne

- KolejkaLista ()
  *Konstruktor.*
- ~KolejkaLista ()
  *Destruktor.*

### Statyczne metody publiczne

- static bool enqueue (KolejkaLista &cel, TYP element)
  *Wstawia element na poczatek kolejki.*
- static bool dequeue (KolejkaLista &cel, TYP &element)
  *Usuwa element z konca kolejki.*
- static bool isempty (KolejkaLista &cel)
  *Sprawdza czy kolejka nie ma zadnych elementow.*
- static int size (KolejkaLista &cel)
  *Sprawdza ile elementow jest w kolejce.*

#### 4.3.1 Opis szczegółowy

Implementacja kolejki na liscie.

#### 4.3.2 Dokumentacja funkcji składowych

##### 4.3.2.1 bool KolejkaLista::dequeue ( KolejkaLista & *cel,* TYP & *element* ) [static]

Usuwa element z konca kolejki.

**Parametry**

| | |
|---|---|
| *cel* | docelowa kolejka. |
| *element* | sciagniety z kolejki. |

**Zwraca**

powodzenie wykonania funkcji.

##### 4.3.2.2 bool KolejkaLista::enqueue ( KolejkaLista & *cel,* TYP *element* ) [static]

Wstawia element na poczatek kolejki.

**Parametry**

| | |
|---|---|
| *cel* | docelowa kolejka. |
| *element* | do wstawienia na kolejke docelowa. |

**Zwraca**

powodzenie wykonania funkcji.

##### 4.3.2.3 bool KolejkaLista::isempty ( KolejkaLista & *cel* ) [static]

Sprawdza czy kolejka nie ma zadnych elementow.

**Parametry**

| | |
|---|---|
| *cel* | docelowkolejka. |

**Zwraca**

true - jesli pusty. false - jesli nie pusty.

##### 4.3.2.4 int KolejkaLista::size ( KolejkaLista & *cel* ) [static]

Sprawdza ile elementow jest w kolejce.

**Parametry**

| | |
|---|---|
| *cel* | docelowa kolejka. |

**Zwraca**

ilosc elementow.

Dokumentacja dla tej klasy została wygenerowana z plików:

- KolejkaLista.h
- KolejkaLista.cpp

## 4.4 Dokumentacja klasy KolejkaTab

Implementacja kolejki na tablicy.

`#include <KolejkaTablica.h>`

### Metody publiczne

- KolejkaTab ()
  *Konstruktor.*
- ~KolejkaTab ()
  *Destruktor.*

### Statyczne metody publiczne

- static bool enqueue (KolejkaTab &cel, TYP element)
  *Wstawia element na poczatek kolejki.*
- static bool dequeue (KolejkaTab &cel, TYP &element)
  *Usuwa element z konca kolejki.*
- static bool isempty (KolejkaTab &cel)
  *Sprawdza czy kolejka nie ma zadnych elementow.*
- static int size (KolejkaTab &cel)
  *Sprawdza ile elementow jest w kolejce.*

### 4.4.1 Opis szczegółowy

Implementacja kolejki na tablicy.

### 4.4.2 Dokumentacja funkcji składowych

#### 4.4.2.1 bool KolejkaTab::dequeue ( KolejkaTab & *cel,* TYP & *element* ) [static]

Usuwa element z konca kolejki.

**Parametry**

| | |
|---|---|
| *cel* | docelowa kolejka. |
| *element* | sciagniety z kolejki. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.4.2.2 bool KolejkaTab::enqueue ( KolejkaTab & *cel,* TYP *element* ) [static]

Wstawia element na poczatek kolejki.

**Parametry**

| | |
|---|---|
| *cel* | docelowa kolejka. |
| *element* | do wstawienia na kolejke docelowa. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.4.2.3 bool KolejkaTab::isempty ( KolejkaTab & *cel* ) [static]

Sprawdza czy kolejka nie ma zadnych elementow.

**Parametry**

| | |
|---|---|
| *cel* | docelowkolejka. |

**Zwraca**

true - jesli pusty. false - jesli nie pusty.

#### 4.4.2.4 int KolejkaTab::size ( KolejkaTab & *cel* ) [static]

Sprawdza ile elementow jest w kolejce.

**Parametry**

| | |
|---|---|
| *cel* | docelowa kolejka. |

**Zwraca**

ilosc elementow.

Dokumentacja dla tej klasy została wygenerowana z plików:

- KolejkaTablica.h
- KolejkaTablica.cpp

## 4.5 Dokumentacja klasy StosLista

Implementacja stosu na liscie.

`#include <StosLista.h>`

**Metody publiczne**

- StosLista ()
  *Konstruktor.*
- ~StosLista ()
  *Destruktor.*

**Statyczne metody publiczne**

- static bool push (StosLista &cel, TYP element)
  *Wstawia element na szczyt stosu.*
- static bool pop (StosLista &cel, TYP &element)
  *Usuwa element ze szczytu stosu.*
- static bool isempty (StosLista &cel)
  *Sprawdza czy stos nie ma zadnych elementow.*
- static int size (StosLista &cel)
  *Sprawdza ile elementow jest na stosie.*

### 4.5.1 Opis szczegółowy

Implementacja stosu na liscie.

#### 4.5.2 Dokumentacja funkcji składowych

##### 4.5.2.1 bool StosLista::isempty ( StosLista & *cel* ) [static]

Sprawdza czy stos nie ma zadnych elementow.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |

**Zwraca**

true - jesli pusty. false - jesli nie pusty.

#### 4.5.2.2 bool StosLista::pop ( StosLista & *cel*, TYP & *element* ) [static]

Usuwa element ze szczytu stosu.

**Parametry**

| | |
|---|---|
| *cel* | docelowey stos. |
| *element* | sciagniety ze stosu. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.5.2.3 bool StosLista::push ( StosLista & *cel*, TYP *element* ) [static]

Wstawia element na szczyt stosu.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |
| *element* | do wstawienia na stos docelowy. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.5.2.4 int StosLista::size ( StosLista & *cel* ) [static]

Sprawdza ile elementow jest na stosie.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |

**Zwraca**

ilosc elementow.

Dokumentacja dla tej klasy została wygenerowana z plików:

- StosLista.h
- StosLista.cpp

## 4.6 Dokumentacja klasy StosTab1

Implementacja stosu na tablicy. Wersja z powiekszeniem tablicy o 1 element przy przekroczeniu.

`#include <StosTab1.h>`

#### Metody publiczne

- StosTab1 ()

  *Konstruktor.*
- ~StosTab1 ()

  *Destruktor usuwa zaalakowana dynamicznie tablice.*

#### Statyczne metody publiczne

- static bool push (StosTab1 &cel, TYP element)

  *Wstawia element na szczyt stosu.*
- static bool pop (StosTab1 &cel, TYP &element)

  *Usuwa element ze szczytu stosu.*
- static bool isempty (StosTab1 &cel)

  *Sprawdza czy stos nie ma zadnych elementow.*
- static int size (StosTab1 &cel)

  *Sprawdza ile elementow jest na stosie.*

### 4.6.1 Opis szczegółowy

Implementacja stosu na tablicy. Wersja z powiekszeniem tablicy o 1 element przy przekroczeniu.

### 4.6.2 Dokumentacja funkcji składowych

#### 4.6.2.1 bool StosTab1::isempty ( StosTab1 & *cel* ) [static]

Sprawdza czy stos nie ma zadnych elementow.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |

**Zwraca**

true - jesli pusty. false - jesli nie pusty.

#### 4.6.2.2 bool StosTab1::pop ( StosTab1 & *cel*, TYP & *element* ) [static]

Usuwa element ze szczytu stosu.

**Parametry**

| | |
|---|---|
| *cel* | docelowey stos. |
| *element* | sciagniety ze stosu. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.6.2.3 bool StosTab1::push ( StosTab1 & *cel*, TYP *element* ) [static]

Wstawia element na szczyt stosu.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |
| *element* | do wstawienia na stos docelowy. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.6.2.4 int StosTab1::size ( StosTab1 & *cel* ) [static]

Sprawdza ile elementow jest na stosie.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |

**Zwraca**

ilosc elementow.

Dokumentacja dla tej klasy została wygenerowana z plików:

- StosTab1.h
- StosTab1.cpp

## 4.7 Dokumentacja klasy StosTabx2

Implementacja stosu na tablicy. Wersja z powiekszeniem tablicy dwukrotnie przy przekroczeniu.

`#include <StosTabx2.h>`

**Metody publiczne**

- StosTabx2 ()
  *Konstruktor.*
- ~StosTabx2 ()
  *Destruktor usuwa zaalakowana dynamicznie tablice.*

**Statyczne metody publiczne**

- static bool push (StosTabx2 &cel, TYP element)
  *Wstawia element na szczyt stosu.*
- static bool pop (StosTabx2 &cel, TYP &element)
  *Usuwa element ze szczytu stosu.*
- static bool isempty (StosTabx2 &cel)
  *Sprawdza czy stos nie ma zadnych elementow.*
- static int size (StosTabx2 &cel)
  *Sprawdza ile elementow jest na stosie.*

### 4.7.1 Opis szczegółowy

Implementacja stosu na tablicy. Wersja z powiekszeniem tablicy dwukrotnie przy przekroczeniu.

#### 4.7.2 Dokumentacja funkcji składowych

##### 4.7.2.1 bool StosTabx2::isempty ( StosTabx2 & *cel* ) [static]

Sprawdza czy stos nie ma zadnych elementow.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |

**Zwraca**

true - jesli pusty. false - jesli nie pusty.

#### 4.7.2.2 bool StosTabx2::pop ( StosTabx2 & *cel*, TYP & *element* ) [static]

Usuwa element ze szczytu stosu.

**Parametry**

| | |
|---|---|
| *cel* | docelowey stos. |
| *element* | sciagniety ze stosu. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.7.2.3 bool StosTabx2::push ( StosTabx2 & *cel*, TYP *element* ) [static]

Wstawia element na szczyt stosu.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |
| *element* | do wstawienia na stos docelowy. |

**Zwraca**

powodzenie wykonania funkcji.

#### 4.7.2.4 int StosTabx2::size ( StosTabx2 & *cel* ) [static]

Sprawdza ile elementow jest na stosie.

**Parametry**

| | |
|---|---|
| *cel* | docelowy stos. |

**Zwraca**

ilosc elementow.

Dokumentacja dla tej klasy została wygenerowana z plików:

- StosTabx2.h
- StosTabx2.cpp

# 5 Dokumentacja plików

## 5.1 Dokumentacja pliku definicje.h

Plik zawiera ogolne instrukcje preprocesora wspolne dla wszystkich plikow zrodlowych.

#### Definicje

- #define **TYP** int

### 5.1.1 Opis szczegółowy

Plik zawiera ogolne instrukcje preprocesora wspolne dla wszystkich plikow zrodlowych.

## 5.2 Dokumentacja pliku KolejkaLista.cpp

```
#include "KolejkaLista.h"
```

## 5.3 Dokumentacja pliku KolejkaLista.h

```
#include "definicje.h"
#include <stdlib.h>
```

#### Komponenty

- struct ElementKolejki

  *Struktura pojedynczego elementu przechowywanego w kolejce.*
- class KolejkaLista

  *Implementacja kolejki na liscie.*

## 5.4 Dokumentacja pliku KolejkaTablica.cpp

```
#include "KolejkaTablica.h"
```

## 5.5 Dokumentacja pliku KolejkaTablica.h

```
#include "definicje.h"
#include <stdlib.h>
```

#### Komponenty

- class KolejkaTab

  *Implementacja kolejki na tablicy.*

## 5.6 Dokumentacja pliku StosLista.cpp

```
#include "StosLista.h"
```

## 5.7 Dokumentacja pliku StosLista.h

```
#include "definicje.h"
#include <stdlib.h>
```

### Komponenty

- struct ElementStosu

  *Struktura pojedynczego elementu przechowywanego w stosie.*
- class StosLista

  *Implementacja stosu na liscie.*

## 5.8 Dokumentacja pliku StosTab1.cpp

```
#include "StosTab1.h"
```

## 5.9 Dokumentacja pliku StosTab1.h

```
#include <stdlib.h>
#include "definicje.h"
```

### Komponenty

- class StosTab1

  *Implementacja stosu na tablicy. Wersja z powiekszeniem tablicy o 1 element przy przekroczeniu.*

## 5.10 Dokumentacja pliku StosTabx2.h

```
#include <stdlib.h>
#include "definicje.h"
```

### Komponenty

- class StosTabx2

  *Implementacja stosu na tablicy. Wersja z powiekszeniem tablicy dwukrotnie przy przekroczeniu.*

## 5.11 Dokumentacja pliku TestStosowIKolejek.cpp

```
#include <iostream>
#include <fstream>
#include <time.h>
#include <stdlib.h>
#include "definicje.h"
#include "StosTab1.h"
#include "StosTabx2.h"
#include "StosLista.h"
#include "KolejkaLista.h"
#include "KolejkaTablica.h"
```

**Definicje**

- #define PLIK_DANYCH "dane3.txt"

  *nazwa pliku z danymi wejsciowymi, takze dla wyjscia generowanych liczb.*
- #define LICZBA_POWTORZEN 10

  *ilosc powtorzen pomiaru czasu dla kazdego rozmiaru problemu.*
- #define LICZBA_WIELKOSCI 9

  *ilosc roznych rozmiarow problemu.*
- #define WIELKOSCI_PROBLEMU {10, 100, 1000, 2000, 3000, 4000, 6000, 8000, 10000}

  *tablica zawierajaca wszystkie mierzone rozmiary problemu.*
- #define WIELKOSC_GENEROWANYCH_DANYCH 10000

  *ilosc danych do wygenerowania.*

**Funkcje**

- double algorytm1 (TYP ∗zrodlo, int rozmiar)

  *Testowany algorytm dla stosu tablicowego z powiekszeniem 1.*
- double algorytm2 (TYP ∗zrodlo, int rozmiar)

  *Testowany algorytm dla stosu tablicowego z powiekszeniem x2.*
- bool wczytaj_dane (const char ∗nazwa_pliku, TYP ∗&tablica, int &rozmiar)

  *wczytuje dane z pliku*
- bool zapisz_dane (const char ∗nazwa_pliku, int ∗col_rozmiar_problemu, double ∗col_czas, int rozmiar)

  *zapisuje dane do pliku .csv dane zawieraja: 1 kolumna: wielkosc problemu 2 kolumna: czas potrzebny do zrealizowanego danego problemu*
- void testuj_algorytm (double(∗algorytm)(TYP ∗zrodlo, int rozmiar), TYP ∗tablica, int rozmiar, const char ∗plik_wyjsciowy)

  *wykonuje testy czasu algorytmu dla przygotowanych parametrow zmierzone czasu zapisuje do pliku*
- void generuj_dane ()

  *generuje dane jesli ich nie ma*
- void sprawdz_poprawnosc ()

  *funkcja sprawdza poprawnosc dzialania struktur danych.*
- int main ()

  *funkcja main*

### 5.11.1 Dokumentacja funkcji

#### 5.11.1.1 double algorytm1 ( TYP ∗ *zrodlo*, int *rozmiar* )

Testowany algorytm dla stosu tablicowego z powiekszeniem 1.

**Parametry**

| | |
|---|---|
| *zrodlo* | tablica danych do wrzucenia na stos |
| *rozmiar* | ilosc danych |

**Zwraca**

czas wykonywania

#### 5.11.1.2 double algorytm2 ( TYP ∗ *zrodlo*, int *rozmiar* )

Testowany algorytm dla stosu tablicowego z powiekszeniem x2.

**Parametry**

| | |
|---|---|
| *zrodlo* | tablica danych do wrzucenia na stos |
| *rozmiar* | ilosc danych |

**Zwraca**

czas wykonywania

#### 5.11.1.3 void testuj_algorytm ( double(∗)(TYP ∗zrodlo, int rozmiar) *algorytm,* TYP ∗ *tablica,* int *rozmiar,* const char ∗ *plik_wyjsciowy* )

wykonuje testy czasu algorytmu dla przygotowanych parametrow zmierzone czasu zapisuje do pliku

∗∗

**Parametry**

| | |
|---|---|
| *double(∗algorytm)* | funkcja z algorytmem do testowania |
| *∗tablica* | dane dla algorytmu |
| *rozmiar* | rozmiar tablicy |
| *∗plik_wyjsciowy* | nazwa pliku do zapisu zmierzonych czasow |

#### 5.11.1.4 bool wczytaj_dane ( const char ∗ *nazwa_pliku,* TYP ∗& *tablica,* int & *rozmiar* )

wczytuje dane z pliku

Format:

liczba_danych

dana1

dana2

.

**Parametry**

| | |
|---|---|
| *∗nazwa_pliku* | nazwa pliku z danymi |
| *∗&tablica* | tablica docelowa (usuwana w przypadku !=NULL) |
| *&rozmiar* | rozmiar tablicy docelowej |

#### 5.11.1.5 bool zapisz_dane ( const char ∗ *nazwa_pliku,* int ∗ *col_rozmiar_problemu,* double ∗ *col_czas,* int *rozmiar* )

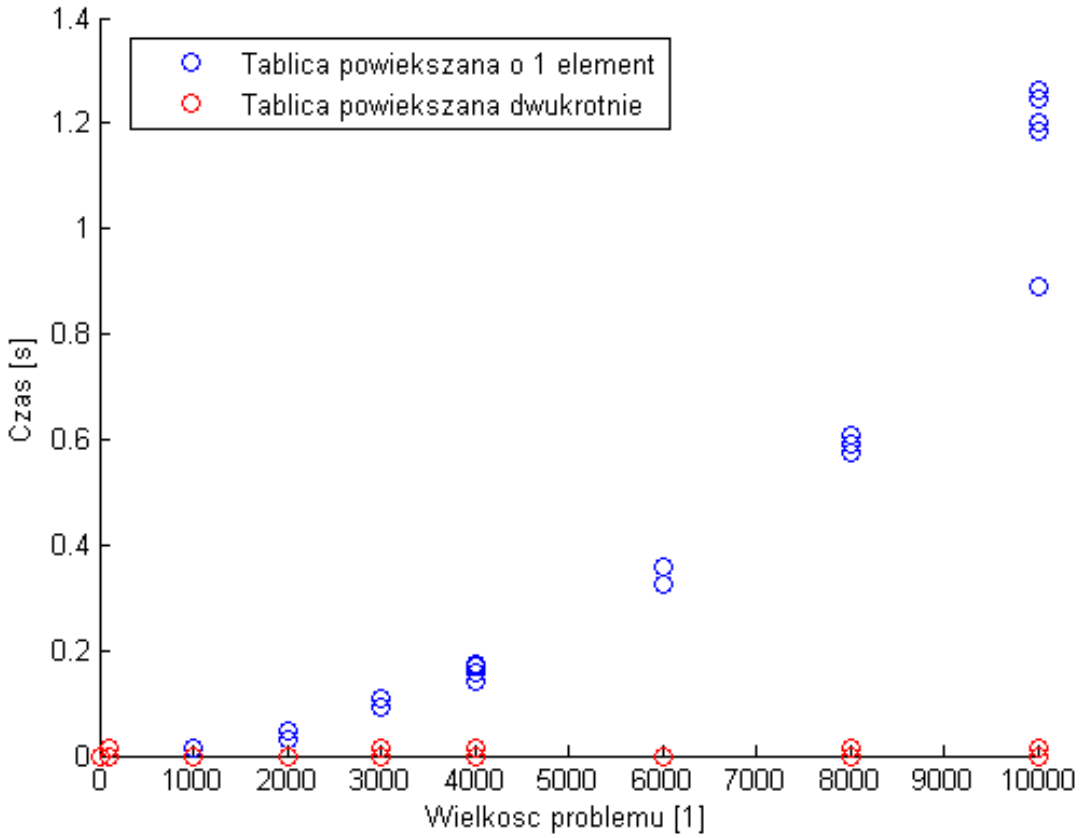
zapisuje dane do pliku .csv dane zawieraja: 1 kolumna: wielkosc problemu 2 kolumna: czas potrzebny do zrealizowanego danego problemu

**Parametry**

| | |
|---|---|
| *∗nazwa_pliku* | nazwa pliku do zapisu |
| *∗col_rozmiar_problemu* | tablica z 1 kolumna |
| *∗col_czas* | druga kolumna |
| *rozmiar* | rozmiar obu tablic |

**Zwraca**

- true sukces
- false blad




Rysunek 1: Wykres pomiaru czasu.

## 6 Sprawozdanie z wykonania programu

Pomiar czasu dla dwóch różnych implementacji stosu na tablicy przedstawiono na wykresie 1. Z wykresu wynika, że implementacja korzystająca z powiększania tablicy zawsze o jeden element ma dużo większą złożoność obliczeniową (prawdopodobnie kwadratową) niż implementacja polegająca na powiększaniu tablicy dwukrotnie.

## Skorowidz