

Program framework benchmarkujacy dla struktury danych Stos

1.1

Wygenerowano przez Doxygen 1.8.9.1

Śr, 25 mar 2015 13:12:53

Spis treści

1	Program framework benchmarkujący dla struktury danych Stos	2
2	Indeks klas	2
2.1	Lista klas	2
3	Indeks plików	2
3.1	Lista plików	2
4	Dokumentacja klas	3
4.1	Dokumentacja szablonu klasy Kolejka< typ >	3
4.1.1	Opis szczegółowy	3
4.1.2	Dokumentacja konstruktora i destruktora	3
4.1.3	Dokumentacja funkcji składowych	4
4.1.4	Dokumentacja atrybutów składowych	4
4.2	Dokumentacja szablonu klasy Lista< typ >	4
4.2.1	Opis szczegółowy	5
4.2.2	Dokumentacja konstruktora i destruktora	5
4.2.3	Dokumentacja funkcji składowych	5
4.2.4	Dokumentacja atrybutów składowych	6
4.3	Dokumentacja szablonu klasy Stos< typ >	6
4.3.1	Opis szczegółowy	7
4.3.2	Dokumentacja konstruktora i destruktora	7
4.3.3	Dokumentacja funkcji składowych	7
4.3.4	Dokumentacja atrybutów składowych	8
5	Dokumentacja plików	8
5.1	Dokumentacja pliku Kolejka.hh	8
5.2	Dokumentacja pliku Lista.hh	8
5.3	Dokumentacja pliku main.cpp	9
5.3.1	Dokumentacja funkcji	9
5.4	Dokumentacja pliku Stoper.hh	9
5.4.1	Dokumentacja funkcji	10
5.5	Dokumentacja pliku Stos.hh	11
6	Zadanie	12
7	Realizacja	12
8	Działanie	12
9	Wyniki	12

10 Komentarz

12

1 Program framework benchmarkujący dla struktury danych Stos

Autor

Wojciech Makuch

Data

16.03.2015

Wersja

1.1 Program przeprowadza operacje zliczania czasu wypełnienia struktury danych stos liczbami pseudolosowymi. Program sprawdza działanie i szybkość obliczeniową struktury danych przy alokacji pamięci zarówno o 1 element i razy 200%. Wyliczony czas podawany z dokładnością do us. Uzyskane dane program zapisuje do pliku o nazwie "Pomiar_czasu3.txt"

2 Indeks klas

2.1 Lista klas

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

Kolejka< typ >

Definicja klasy **Kolejka** Zbudowana na tablicy posiada indeksy pokazujące na początek i na koniec kolejki. Zbudowana na szablonie

3

Lista< typ >

Definicja klasy **Lista** Przechowuje obiekt oraz wskaźnik na następny i pole rozmiar. Zbudowana na szablonie

4

Stos< typ >

Definicja klasy **Stos** zdefiniowany za pomocą tablicy. Klasa zbudowana na szablonie

6

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Kolejka.hh

Definicja struktury danych **Kolejka**

8

Lista.hh

Definicja struktury danych **Lista**

8

main.cpp

9

Stoper.hh

Definicje funkcji zliczających czas operacji wypełnienia struktur danych

9

Stos.hh

Definicja struktury danych **Stos**

11

4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy Kolejka< typ >

definicja klasy `Kolejka` Zbudowana na tablicy posiada indeksy pokazujące na początek i na koniec kolejki. Zbudowana na szablonie.

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

Metody publiczne

- `Kolejka` (int *ilosc*)
definicja konstruktora z jednym parametrem
- `Kolejka` ()
definicja konstruktora bezparametrycznego Zeruje rozmiar, ustawia wskaznik tablicy na NULL, zeruje indeksy.
- `~Kolejka` ()
definicja destruktoru Zwalnia zaalokowana pamiec tablicy. Zeruje indeksy i rozmiar.
- int `size` () const
definicja metody size
- void `enqueue` (typ *element*)
definicja metody enqueue
- typ `dequeue` ()
definicja metody dequeue

Atrybuty prywatne

- int *f*
- int *r*
- typ * *tab*
- int *rozmiar*

4.1.1 Opis szczegółowy

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

Definicja w linii 17 pliku Kolejka.hh.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 template<typename typ > Kolejka< typ >::Kolejka (int *ilosc*)

max rozmiar tablicy

Parametry

<i>[ilosc]</i>	rozmiar alokowanej tablicy Alokuje tablice o zadany rozmiarze. Ustawia indeksy na 0.
----------------	--

Definicja w linii 38 pliku Kolejka.hh.

4.1.2.2 template<typename typ > Kolejka< typ >::Kolejka ()

Definicja w linii 51 pliku Kolejka.hh.

4.1.2.3 template<typename typ > Kolejka< typ >::~~Kolejka ()

Definicja w linii 64 pliku Kolejka.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 `template<typename typ > typ Kolejka< typ >::dequeue ()`

Zwraca

element z początku kolejki
0 i wyświetla komunikat gdy kolejka jest pusta. zmienia położenie indeksu początku.

Definicja w linii 126 pliku Kolejka.hh.

4.1.3.2 `template<typename typ > void Kolejka< typ >::enqueue (typ element)`

Parametry

<i>[element]</i>	dodawany element Dodaje element na koniec kolejki. Gdy kolejka jest pełna, powiększa tablicę o 5 i przekopiuje elementy. zmienia położenie indeksu końca.
------------------	---

Definicja w linii 92 pliku Kolejka.hh.

4.1.3.3 `template<typename typ > int Kolejka< typ >::size () const`

Zwraca

rozmiar ilości danych przechowywanych w tablicy.
0, gdy kolejka jest pusta.

Definicja w linii 76 pliku Kolejka.hh.

4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 `template<typename typ > int Kolejka< typ >::f [private]`

Definicja w linii 18 pliku Kolejka.hh.

4.1.4.2 `template<typename typ > int Kolejka< typ >::r [private]`

początek

Definicja w linii 19 pliku Kolejka.hh.

4.1.4.3 `template<typename typ > int Kolejka< typ >::rozmiar [private]`

przechowywane elementy

Definicja w linii 21 pliku Kolejka.hh.

4.1.4.4 `template<typename typ > typ* Kolejka< typ >::tab [private]`

koniec

Definicja w linii 20 pliku Kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Kolejka.hh](#)

4.2 Dokumentacja szablonu klasy Lista< typ >

definicja klasy [Lista](#) Przechowuje obiekt oraz wskaźnik na następny i pole rozmiar. Zbudowana na szablonie.

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

Metody publiczne

- Lista ()
definicja konstruktora bezparametrycznego Zeruje rozmiar, ustawia wskaznik na NULL.
- ~Lista ()
definicja destruktor Zeruje rozmiar, Kasuje wszystkie obiektry/elementy.
- void push (typ element)
definicja metody push
- typ pop ()
definicja metody pop
- int size () const
definicja metody size

Atrybuty prywatne

- Lista< typ > * nastepny
- typ dane
- int rozmiar

4.2.1 Opis szczegółowy

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

Definicja w linii 15 pliku Lista.hh.

4.2.2 Dokumentacja konstruktora i destruktor

4.2.2.1 template<typename typ > Lista< typ >::Lista ()

ilosc elementow/obiektow

Definicja w linii 33 pliku Lista.hh.

4.2.2.2 template<typename typ > Lista< typ >::~~Lista ()

Definicja w linii 94 pliku Lista.hh.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 template<typename typ > typ Lista< typ >::pop ()

Zwraca

usuwany element Ustawia wskaznik na poprzedni element zwraca i kasuje ostatni element.
0 i wyswietla komunikat gdy lista jest pusta.

Definicja w linii 62 pliku Lista.hh.

4.2.3.2 template<typename typ > void Lista< typ >::push (typ element)

Parametry

<i>[element]</i>	dodawany element na koniec listy Zwiększa rozmiar, alokuje pamięć, przypisuje element do pola klasy.
------------------	--

Definicja w linii 45 pliku Lista.hh.

4.2.3.3 `template<typename typ> int Lista< typ >::size () const`

Zwraca

ilość elementów przechowywanych na liście.

Definicja w linii 83 pliku Lista.hh.

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 `template<typename typ> typ Lista< typ >::dane [private]`

wskaznik na następny obiekt/element

Definicja w linii 17 pliku Lista.hh.

4.2.4.2 `template<typename typ> Lista<typ>* Lista< typ >::nastepny [private]`

Definicja w linii 16 pliku Lista.hh.

4.2.4.3 `template<typename typ> int Lista< typ >::rozmiar [private]`

przechowywana informacja/obiekt/element

Definicja w linii 18 pliku Lista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Lista.hh](#)

4.3 Dokumentacja szablonu klasy Stos< typ >

definicja klasy [Stos](#) zdefiniowany za pomocą tablicy. Klasa zbudowana na szablonie.

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

Metody publiczne

- [Stos](#) (typ p)
definicja konstruktora z jednym parametrem
- [Stos](#) ()
definicja konstruktora bezparametrycznego zeruje rozmiar, przypisuje NULL do wskaźników.
- [~Stos](#) ()
definicja destruktoru Zwalnia pamięć, zeruje rozmiar.
- void [push](#) (typ element)
definicja metody push
- void [push200](#) (typ element)
definicja metody push200
- typ [pop](#) ()
definicja metody pop zmniejsza licznik elementów o 1, jeśli licznik jest mniejszy od rozmiaru alokowanej pamięci 25% rozmiar jest zmniejszany o te 25%
- int [size](#) () const
definicja metody size

- int `size_licznik` () const
definicja metody `size_licznik`

Atrybuty prywatne

- int `rozmiar`
- int `licznik`
- typ * `tab`

4.3.1 Opis szczegółowy

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

Definicja w linii 16 pliku Stos.hh.

4.3.2 Dokumentacja konstruktora i destruktor

4.3.2.1 `template<typename typ > Stos< typ >::Stos (typ p)`

alokowana pamiec

Parametry

<code>[p]</code>	rozmiar ilosci alokowanej pamieci alokuje pamiec o zadanym rozmiarze
------------------	--

Definicja w linii 38 pliku Stos.hh.

4.3.2.2 `template<typename typ > Stos< typ >::~Stos ()`

Definicja w linii 50 pliku Stos.hh.

4.3.2.3 `template<typename typ > Stos< typ >::~~Stos ()`

Definicja w linii 62 pliku Stos.hh.

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 `template<typename typ > typ Stos< typ >::pop ()`

Zwraca

usuwany element
0 i wyswietla komunikat, kiedy stos jest pusty.

Definicja w linii 128 pliku Stos.hh.

4.3.3.2 `template<typename typ > void Stos< typ >::push (typ element)`

Parametry

<code>[element]</code>	dodany element na koniec stosu zwiększa licznik o 1, alokuje nowa tablice, kopiuje zawartosc starej do nowej, kladzie element na ostatniej pozycji, realokuje i przekopiuje zawartosc do pierwotnej tablicy, usuwa tablice pomocnicza.
------------------------	--

Definicja w linii 78 pliku Stos.hh.

4.3.3.3 `template<typename typ > void Stos< typ >::push200 (typ element)`

Parametry

<i>[element]</i>	dodawany element na koniec stosu zwiększa rozmiar razy 200%, dodaje element na stos zwiększa licznik ilości elementów realokuje i przekopiuje wartości tablic
------------------	---

Definicja w linii 100 pliku Stos.hh.

4.3.3.4 `template<typename typ > int Stos< typ >::size () const`

Zwraca

rozmiar stosu

Definicja w linii 149 pliku Stos.hh.

4.3.3.5 `template<typename typ > int Stos< typ >::size_licznik () const`

Zwraca

licznik elementów na stosie

Definicja w linii 159 pliku Stos.hh.

4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 `template<typename typ> int Stos< typ >::licznik [private]`

rozmiar stosu

Definicja w linii 18 pliku Stos.hh.

4.3.4.2 `template<typename typ> int Stos< typ >::rozmiar [private]`

Definicja w linii 17 pliku Stos.hh.

4.3.4.3 `template<typename typ> typ* Stos< typ >::tab [private]`

ilość zapisanych danych na stosie

Definicja w linii 19 pliku Stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Stos.hh](#)

5 Dokumentacja plików

5.1 Dokumentacja pliku Kolejka.hh

definicja struktury danych [Kolejka](#)

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

5.2 Dokumentacja pliku Lista.hh

definicja struktury danych [Lista](#)

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [Lista< typ >](#)
definicja klasy [Lista](#) Przechowuje obiekt oraz wskaźnik na następny i pole rozmiar. Zbudowana na szablonie.

5.3 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include "Lista.hh"
#include "Stos.hh"
#include "Kolejka.hh"
#include "Stoper.hh"
```

Wykres zależności załączania dla main.cpp:

Funkcje

- int [main](#) ()

5.3.1 Dokumentacja funkcji

5.3.1.1 int main ()

Definicja w linii 24 pliku main.cpp.

5.4 Dokumentacja pliku Stoper.hh

definicje funkcji zliczających czas operacji wypełnienia struktur danych

```
#include <windows.h>
#include <ctime>
#include <fstream>
```

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

Funkcje

- LARGE_INTEGER [startTimer](#) ()
definicja funkcji [startTimer](#) Rozpoczyna pomiar czasu Funkcja pobrana ze strony <http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/>
- LARGE_INTEGER [endTimer](#) ()
definicja funkcji [endTimer](#) Konczy pomiar czasu Funkcja pobrana ze strony <http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/>
- template<typename typ >
double [licz](#) (typ obiekt, int N)
definicja funkcji [benchamarkujacej](#) [licz](#) funkcja nie przeznaczona dla struktury danych [Kolejka](#)
- template<typename typ >
double [licz200](#) (typ obiekt, int N)
definicja funkcji [benchamarkujacej](#) [licz200](#) Funkcja przeznaczona dla struktury danych [stos](#), działającej przy alokowaniu pamięci razy 200%. Funkcja nie przeznaczona dla struktury danych [Kolejka](#) i lista oraz stos działającej w sposób alokowania pamięci o 1 element.
- template<typename typ >
double [liczKol](#) (typ obiekt, int N)

definicja funkcji `liczKol` Funkcja Benchmarkujaca przeznaczona tylko dla struktury danych `Kolejka` Posiada atrybuty takie same jak funkcja `licz`

- `template<typename typ >`
`double zliczaj (typ obiekt, int N)`

definicja funkcji `zliczaj` Alternatywna funkcja benchmarkujaca nie przeznaczona dla struktury danych `Kolejka`

5.4.1 Dokumentacja funkcji

5.4.1.1 `LARGE_INTEGER endTimer ()`

Definicja w linii 31 pliku `Stoper.hh`.

5.4.1.2 `template<typename typ > double licz (typ obiekt, int N)`

Parametry

<code>[obekt]</code>	struktura danych, dla ktorej zostana przeprowadzone obliczenia
<code>[N]</code>	ilosc elementow, ktorymi zostanie wypelniona struktura danych

Zwraca

czas wypelnienia struktury danych liczbami pseudolosowymi Funkcja pobrana ze strony <http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/>

Definicja w linii 49 pliku `Stoper.hh`.

5.4.1.3 `template<typename typ > double licz200 (typ obiekt, int N)`

Parametry

<code>[obekt]</code>	struktura danych, dla ktorej zostana przeprowadzone obliczenia
<code>[N]</code>	ilosc elementow, ktorymi zostanie wypelniona struktura danych

Zwraca

czas wypelnienia struktury danych liczbami pseudolosowymi Funkcja pobrana ze strony <http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/>

Definicja w linii 76 pliku `Stoper.hh`.

5.4.1.4 `template<typename typ > double liczKol (typ obiekt, int N)`

Definicja w linii 97 pliku `Stoper.hh`.

5.4.1.5 `LARGE_INTEGER startTimer ()`

Definicja w linii 17 pliku `Stoper.hh`.

5.4.1.6 `template<typename typ > double zliczaj (typ obiekt, int N)`

Parametry

<code>[obekt]</code>	struktura danych
<code>[N]</code>	ilosc liczb pseudolosowych, ktorymi zostanie wypelniona struktura danych

Zwraca

czas wykonania operacji podany w ms. Funkcja nie wykorzystana w programie!!

Definicja w linii 122 pliku `Stoper.hh`.

5.5 Dokumentacja pliku Stos.hh

definicja struktury danych [Stos](#)

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:

Komponenty

- class [Stos](#)< [typ](#) >

definicja klasy [Stos](#) zdefiniowany za pomocą tablicy. Klasa zbudowana na szablonie.

Laboratorium 3 - Sprawozdanie

Wojciech Makuch

25 marca 2015

6 Zadanie

Program framework benchmarkujący dla zaimplementowanej struktury danych Stos.

7 Realizacja

Program zawiera 3 struktury danych. Każda z nich zawiera 3 podstawowe metody: połóż element, zdejmij element, zwróć rozmiar. Struktura Stos zbudowana za pomocą tablicy z realokacją pamięci, Lista ze wskaźnikiem na następny element, oraz Kolejka z indeksami na pierwszy i ostatni element. Wszystkie struktury danych działają prawidłowo. Ponadto program zawiera funkcje wypełniającą struktury liczbami pseudolosowymi oraz zliczającą czas dla przeprowadzenia testów złożoności obliczeniowej ww. struktur.

8 Działanie

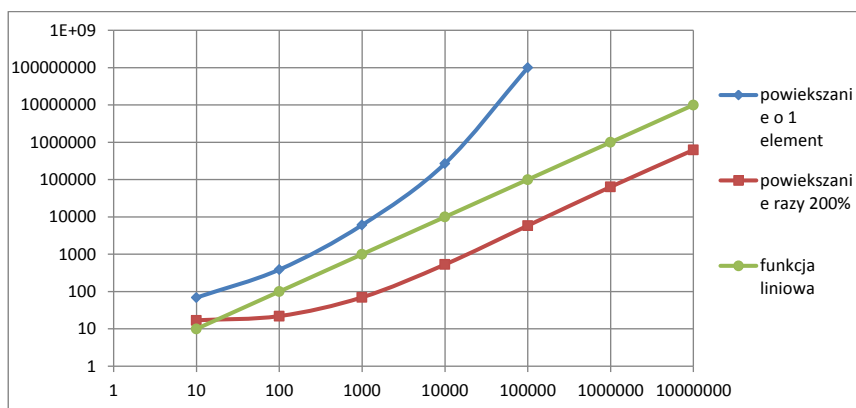
Główna funkcja programu działa tylko na strukturze typu Stos. Testuje jego 2 metody `push()` oraz `push200()`. Metoda `push()` polega na powiększaniu alokowanej pamięci o 1 element, natomiast `push200()` polega na alokowaniu pamięci razy 200%. Działanie programu polega na zliczeniu czasu wypełniania tej struktury liczbami pseudolosowymi oraz zapisania wyników do pliku o nazwie *Pomiar_czasu2.txt*.

9 Wyniki

Podczas alokowania pamięci struktura typu stos obsługiwana przez metodę `push()` może alokować pamięć na maksimum 10^5 elementów. Natomiast dzięki metodzie `push200()` rozmiar ten zostaje zwiększony do 10^7 . Ponadto można zauważyć dłuższy czas destrukcji struktury zaalokowanej przez `push()`. Z danych dostarczonych przez program wynika, że metoda `push200()` działa o wiele szybciej i jest w stanie zaalokować więcej pamięci. Na Rys 1. pokazano w skali logarytmicznej wykres zależności ilości elementów od czasu potrzebnego na wypełnienie nimi struktury. Z wykresu można zauważyć, że złożoność obliczeniowa jest w przybliżeniu liniowa, czyli $O(n)$. Ponadto złożoność metody `push()` rośnie o wiele szybciej, co znaczy, że program ma gorszą złożoność obliczeniową. Dodatkowo na wykresie zaznaczono dla porównania przebieg idealnej funkcji liniowej. Widać, że metoda `push200()` ma nawet lepszą złożoność niż $O(n)$. :)

10 Komentarz

Do utworzenia dokumentacji wykorzystano system Doxygen. Funkcja pomiaru czasu dla systemu Windows pobrana ze strony dr. J. Mierzwy. Program skompilowano w środowisku Code::Blocks. Do stworzenia wykresu posłużono się pakietem MS Excel, sprawozdanie napisano używając systemu \LaTeX .



Rysunek 1: Wykres złożoności obliczeniowej