Program framework benchmarkujacy dla struktur danych Stos, Lista i Kolejka - Laboratorium 2

Wygenerowano przez Doxygen 1.8.9.1

Śr, 18 mar 2015 11:45:35

Spis treści

1 Program framework benchmarkujacy dla struktur danych Stos, Lista i Kolejka 1

2 Indeks klas 2

2.1 Lista klas . . . 2

3 Dokumentacja klas 2

3.1 Dokumentacja szablonu klasy Kolejka< typ > . . . 2

3.1.1 Opis szczegółowy . . . 3

3.1.2 Dokumentacja konstruktora i destruktora . . . 3

3.1.3 Dokumentacja funkcji składowych . . . 3

3.1.4 Dokumentacja atrybutów składowych . . . 4

3.2 Dokumentacja szablonu klasy Lista< typ > . . . 4

3.2.1 Opis szczegółowy . . . 5

3.2.2 Dokumentacja konstruktora i destruktora . . . 5

3.2.3 Dokumentacja funkcji składowych . . . 5

3.2.4 Dokumentacja atrybutów składowych . . . 5

3.3 Dokumentacja szablonu klasy Stos< typ > . . . 6

3.3.1 Opis szczegółowy . . . 6

3.3.2 Dokumentacja konstruktora i destruktora . . . 6

3.3.3 Dokumentacja funkcji składowych . . . 7

3.3.4 Dokumentacja atrybutów składowych . . . 7

4 Zadanie 8

5 Realizacja 8

6 Działanie 8

7 Wyniki 8

8 Komentarz 8

1 Program framework benchmarkujacy dla struktur danych Stos, Lista i Kolejka

Autor

Wojciech Makuch

Data

16.03.2015

Wersja

1.0 Program przeprowadza operacje zliczania czasu wypelnienia trzech struktur danych liczbami pseldolosowymi. Dla Stosu: 10-10e05 elementow. Dla Listy 10-10e07 elementow. Dla Kolejki 10-10e04 elementow. Wyliczony czas podawany z dokladnoscia do 10e-03 ms Uzyskane dane program zapisuje do pliku o nazwie "Pomiar_czasu2.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 pokazujace na poczatek i na koniec kolejki. Zbudowana na szablonie **2**

Lista< typ >
Definicja klasy Lista Przechowuje obiekt oraz wskaznik na nastepny i pole rozmiar. Zbudowana na szablonie **4**

Stos< typ >
Definicja klasy Stos zedfiniowany za pomoca tablicy. Klasa zbudowana na szablonie **6**

3 Dokumentacja klas

3.1 Dokumentacja szablonu klasy Kolejka< typ >

definicja klasy Kolejka Zbudowana na tablicy posiada indeksy pokazujace na poczatek 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 destruktora 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

3.1.1 Opis szczegółowy

template<typename typ>class Kolejka< typ >

definicja klasy Kolejka Zbudowana na tablicy posiada indeksy pokazujace na poczatek i na koniec kolejki. Zbudowana na szablonie.

Definicja w linii 16 pliku Kolejka.hh.

3.1.2 Dokumentacja konstruktora i destruktora

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

definicja konstruktora z jednym parametrem

max rozmiar tablicy

Parametry

[ilosc]	rozmiar alokowanej tablicy Alokuje tablice o zadanym rozmiarze. Ustawia indeksy na 0.

Definicja w linii 37 pliku Kolejka.hh.

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

definicja konstruktora bezparametrycznego Zeruje rozmiar, ustawia wskaznik tablicy na NULL, zeruje indeksy.

Definicja w linii 50 pliku Kolejka.hh.

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

definicja destruktora Zwalnia zaalokowana pamiec tablicy. Zeruje indeksy i rozmiar.

Definicja w linii 63 pliku Kolejka.hh.

3.1.3 Dokumentacja funkcji składowych

3.1.3.1 template<typename typ > typ Kolejka< typ >::dequeue ( )

definicja metody dequeue

Zwraca

element z poczatku kolejki
0 i wyswietla komunikat gdy kolejka jest pusta. zmienia polozenie indeksu poczatku.

Definicja w linii 125 pliku Kolejka.hh.

3.1.3.2 template<typename typ > void Kolejka< typ >::enqueue (typ *element*)

definicja metody enqueue

Parametry

[element]	dodawany element Dodaje element na koniec kolejki. Gdy kolejka jest pelna, powieksza tablice o 5 i przekopiowywuje elementy. zmienia polozenie indeksu konca.

Definicja w linii 91 pliku Kolejka.hh.

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

definicja metody size

Zwraca

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

Definicja w linii 75 pliku Kolejka.hh.

3.1.4 Dokumentacja atrybutów składowych

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

Definicja w linii 17 pliku Kolejka.hh.

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

poczatek

Definicja w linii 18 pliku Kolejka.hh.

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

przechowywane elementy

Definicja w linii 20 pliku Kolejka.hh.

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

koniec

Definicja w linii 19 pliku Kolejka.hh.

3.2 Dokumentacja szablonu klasy Lista< typ >

definicja klasy Lista Przechowuje obiekt oraz wskaznik na nastepny i pole rozmiar. Zbudowana na szablonie.

`#include <Lista.hh>`

Metody publiczne

- Lista ()

  definicja konstruktora bezparametrycznego Zeruje rozmiar, ustawia wskaznik na NULL.
- ~Lista ()

  definicja destruktora Zeruje rozmiar, Kasuje wszystkie obiektry/elementy.
- void push (typ element)

  definicja metody push
- typ pop ()

  definicja metody pop
- int size () const

  deinicja metody size

Atrybuty prywatne

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

3.2.1 Opis szczegółowy

template<typename typ>class Lista< typ >

definicja klasy Lista Przechowuje obiekt oraz wskaznik na nastepny i pole rozmiar. Zbudowana na szablonie.

Definicja w linii 15 pliku Lista.hh.

3.2.2 Dokumentacja konstruktora i destruktora

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

definicja konstruktora bezparametrycznego Zeruje rozmiar, ustawia wskaznik na NULL.

ilosc elementow/obiektow

Definicja w linii 33 pliku Lista.hh.

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

definicja destruktora Zeruje rozmiar, Kasuje wszystkie obiektry/elementy.

Definicja w linii 94 pliku Lista.hh.

3.2.3 Dokumentacja funkcji składowych

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

definicja metody 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.

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

definicja metody push

Parametry

[element]	dodawany element na koniec listy Zwieksza rozmiar, alokuje pamiec, przypisuje element do pola klasy.

Definicja w linii 45 pliku Lista.hh.

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

deinicja metody size

Zwraca

ilosc elementow przechowywanych na liscie.

Definicja w linii 83 pliku Lista.hh.

3.2.4 Dokumentacja atrybutów składowych

3.2.4.1 template<typename typ> typ Lista< typ >::dane [private]

wskaznik na nastepny obiekt/element

Definicja w linii 17 pliku Lista.hh.

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

Definicja w linii 16 pliku Lista.hh.

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

przechowywana informacja/obiekt/element

Definicja w linii 18 pliku Lista.hh.

3.3 Dokumentacja szablonu klasy Stos< typ >

definicja klasy Stos zedfiniowany za pomoca 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 wskaznikow.
- ~Stos ()
  definicja destruktora Zwalnia pamiec, zeruje rozmiar.
- void push (typ element)
  definicja metody push
- typ pop ()
  definicja metody pop zmmniejsza rozmiar o 1,
- int size () const
  definicja metody size

Atrybuty prywatne

- int rozmiar
- typ * tab

3.3.1 Opis szczegółowy

template<typename typ>class Stos< typ >

definicja klasy Stos zedfiniowany za pomoca tablicy. Klasa zbudowana na szablonie.

Definicja w linii 17 pliku Stos.hh.

3.3.2 Dokumentacja konstruktora i destruktora

3.3.2.1 template<typename typ > Stos< typ >::Stos (typ *p*)

definicja konstruktora z jednym parametrem

alokowana pamiec

Parametry

[p]	rozmiar ilosci alokowanej pamieci alokuje pamiec o zadanym rozmiarze

Definicja w linii 36 pliku Stos.hh.

3.3.2.2 template<typename typ > Stos< typ >::Stos ( )

definicja konstruktora bezparametrycznego zeruje rozmiar, przypisuje NULL do wskaznikow.

Definicja w linii 47 pliku Stos.hh.

3.3.2.3 template<typename typ > Stos< typ >::~Stos ( )

definicja destruktora Zwalnia pamiec, zeruje rozmiar.

Definicja w linii 58 pliku Stos.hh.

3.3.3 Dokumentacja funkcji składowych

3.3.3.1 template<typename typ > typ Stos< typ >::pop ( )

definicja metody pop zmmniejsza rozmiar o 1,

Zwraca

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

Definicja w linii 92 pliku Stos.hh.

3.3.3.2 template<typename typ > void Stos< typ >::push (typ *element*)

definicja metody push

Parametry

[element]	dodany element na koniec stosu zwieksza rozmiar o 1, alokuje nowa tablice, kopiuje zawartosc starej do nowej, kladzie element na ostatniej pozycji, realokuje i przekopiowuje zawartosc do pierwotenj tablicy, usuwa tablice pomocnicza.

Definicja w linii 73 pliku Stos.hh.

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

definicja metody size

Zwraca

rozmiar stosu

Definicja w linii 111 pliku Stos.hh.

3.3.4 Dokumentacja atrybutów składowych

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

Definicja w linii 18 pliku Stos.hh.

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

rozmiar stosu

Definicja w linii 19 pliku Stos.hh.

Laboratorium 2 - Sprawozdanie

Wojciech Makuch

18 marca 2015

4 Zadanie

Program framework benchmarkujacy dla zaimpleentowanych struktur Stos, Lista, Kolejka.

5 Realizacja

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

6 Działanie

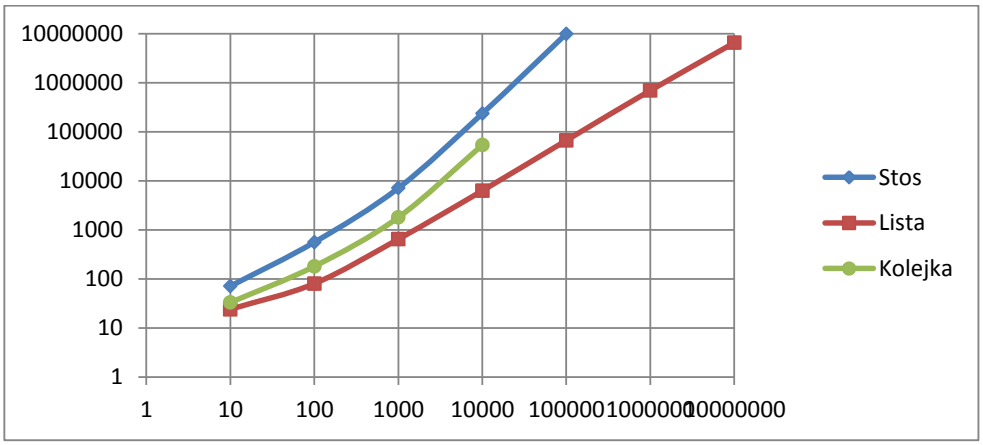
Głowna funkcja programu kolejno tworzy struktury Stos, Lista, Koljaka(a po zakończeniu operacji na nich, zwalnia pamięć), następnie za pomącą funkcji zliczającej czas wykonuje testy złożonosci obliczeniowej. Uzyskae wyniki program wyświetla na ekranie oray zapisuje do pliku o nazwie *Pomiar_czasu2.txt.*

7 Wyniki

Najszybsze działanie algorytmu wypełniania struktur liczbami pesełdolosowymi wykonuje lista, następnie Kolajka, a na samym konsu Stos. Lista ponadto może pomieścić najwięcej elementów, najniej - Kolejka. Na wykresie w skali logarytmicznej pokazano zależność wykonanym operacji do czasu. Widać, że krzywe mozna przybliżyć prostymi, z czego wniosek, że złożoność obliczeniowa wynosi O(n).

8 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 stworzonia wykresu posłużono się pakietem MS Excel, sprawozdanie napisano uzywając systemu LaTeX.



Rysunek 1: Wykres złożoności obliczeniowej.