Pomiar Struktur Danych

Generated by Doxygen 1.8.6

Thu Mar 19 2015 11:49:42

Contents

Chapter 1

Main Page

Czas wykonywania algorytmu wykonującego dodawanie elementów do podatawowych struktour danych

Program realizuje operacje dodawania n liczby elementów do listy, kolejki i stosu i mierzy czas tych operacji

Author

Mateusz Bencer

Date

2015.03.19

Version

1.0

Mail:

209360@pwr.wroc.edu.pl

Wszystkie wykorzystane przeze mnie struktury danych, tj. kolejka oraz stos opierały się na implementacji listy po której dziedziczyły. Wynikało to z uniwersalności listy dwukierunkowej dającej możliwość dodawanie/ściągania elementów z początku oraz końca listy. W związku z takim podejściem wyniki czasowe dodawania elementów (przyjąłem inty) do struktur są bardzo podobne, co potwierdził dobitnie pomiar czasu. Wszystkie zaimplementowane przeze mnie struktury danych są bardzo ogólne (oparte na szablonach i dziedziczeniu) w związku z czym mają szeroką gamme zastosowań. Dla ilości danych powyżej 10 do pot. 7 terminal ubuntu automatycznie zabijał proces.
Wyniki otrzymane na moim komputerze:
Lista:

$$10^2 : 0.046ms \tag{1.1}$$

$$10^3 : 0.558ms \tag{1.2}$$

$$10^6 : 142.849ms \tag{1.3}$$

$$10^7 : 1341.57ms \tag{1.4}$$

Kolejka:

$$10^2 : 0.037ms \tag{1.5}$$

$$10^3 : 0.572ms \tag{1.6}$$

$$10^6 : 136.441ms \tag{1.7}$$

$$10^7 : 1359.62ms \tag{1.8}$$

Stos:

$$10^2 : 0.031ms \tag{1.9}$$

$$10^3 : 0.525ms \tag{1.10}$$

$$10^6 : 133.504ms \tag{1.11}$$

$$10^7 : 1332.96ms \tag{1.12}$$

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Element< T > Class Template Reference

Klasa reprezentująca abstrakcyjny "pojemnik na dane".

`#include <Element.h>`

Public Member Functions

- Element (T *data)

  Konstruktor zapamiętujący adres przechowywanego obiektu.
- void setData (const T *data)

  setter do przechowywanej danej
- T * getData () const

  getter do przechowywanej danej
- Element< T > * next () const

  zwraca wskaźnik do kolejnego elementu na liście
- Element< T > * prev () const

  zwraca wskaźnik do poprzedniego elementu na liście
- void setNext (Element< T > *next)

  ustawia kolejny element listy
- void setPrev (Element< T > *prev)

  ustawia poprzedni element listy
- ~Element ()

  zwalnianie elementu przechowywanego przez klase Element

5.1.1 Detailed Description

template<class T>class Element< T >

Klasa reprezentująca abstrakcyjny "pojemnik na dane".

Klasa może przechowywać dane zdeterminowane przez typ szablonu, zawiera wskaźnik do kolejnego i poprzedniego elementu (implementacja elementu listy).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 template<typename T > Element< T >::Element (T * *data*)

Konstruktor zapamiętujący adres przechowywanego obiektu.

Parameters

data	obiekt/zmienna do przechowania

5.1.2.2 template<typename T > Element< T >::~Element ( )

zwalnianie elementu przechowywanego przez klase Element

Destruktor zapewniający zwalnianie elementu przechowywanego przez klase Element

5.1.3 Member Function Documentation

5.1.3.1 template<typename T > T * Element< T >::getData ( ) const

getter do przechowywanej danej

Metoda do pobrania wskaźnka przechowywanego danej

Returns

wskaźnik do zmiennej przechowywanej przez klase

5.1.3.2 template<typename T > Element< T > * Element< T >::next ( ) const

zwraca wskaźnik do kolejnego elementu na liście

Returns

wskaźnik do kolejnego elementu na liście

5.1.3.3 template<typename T > Element< T > * Element< T >::prev ( ) const

zwraca wskaźnik do poprzedniego elementu na liście

Returns

wskaźnik do poprzedniego elementu na liście

5.1.3.4 template<typename T > void Element< T >::setData (const T * *data*)

setter do przechowywanej danej

Metoda do ustawiania danej przechowywanej przez klase

obiekt/zmienna, która będzie przechowywana

5.1.3.5 template<typename T > void Element< T >::setNext (Element< T > * *next*)

ustawia kolejny element listy

Parameters

next	kolejny element listy

5.1.3.6 template<typename T > void Element< T >::setPrev (Element< T > * *prev*)

ustawia poprzedni element listy

Parameters

prev	poprzedni element listy

The documentation for this class was generated from the following file:

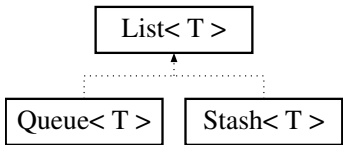
- inc/Element.h

5.2 List< T > Class Template Reference

Klasa reprezentująca podstawy konterner danych z którego korzystają inne - Listę

`#include <List.h>`

Inheritance diagram for List< T >:



Public Types

- enum Direction { **Front**, **Back** }

  Enumerator przekazywany do funckji push w celu określenia, czy umieszczamy element na początku lub na końcu listy [na początku (Front), czy na końcu (Back), None - domyślne dla kontenera].

Public Member Functions

- List ()

  Konstruktor zerujący pola klasy i przydzielający pamięć na _head i _tail.
- Element< T > * pop (Direction dir)

  zwraca element z początku(zależy od użytej struktury danych) listy
- void push (Element< T > *elem, Direction dir)

  dodaje element na początek/koniec(zależy od implementacji) listy
- unsigned int size ()

  zwraca rozmiar użytej struktury danych
- unsigned short isEmpty ()

  zwraca 1, gdy kontener jest pusty, 0 - gdy jest już jakiś element
- virtual ~List ()

  wirtualny destruktor czyszczący liste

5.2.1 Detailed Description

template<class T>class List< T >

Klasa reprezentująca podstawy konterner danych z którego korzystają inne - Listę

Klasa reprezentująca podstawy kontener - listę. Jest to podstawowa implementacja listy stanowiąca klase bazową dla listy, stosu i kolejki

5.2.2 Constructor & Destructor Documentation

5.2.2.1 template<typename T > List< T >::~List ( ) [virtual]

wirtualny destruktor czyszczący liste

Destruktor usuwawa wszystkie elementy z listy

5.2.3 Member Function Documentation

5.2.3.1 template<typename T > unsigned short List< T >::isEmpty ( )

zwraca 1, gdy kontener jest pusty, 0 - gdy jest już jakiś element

Returns

zwraca informacje, czy w kontenerze są już jakieś elementy

5.2.3.2 template<typename T > Element< T > * List< T >::pop (Direction *dir*)

zwraca element z początku(zależy od użytej struktury danych) listy

Parameters

dir	określa czy zdjąć element z początku (Front), czy z końca (Back) listy

Returns

element będący na początku/końcu listy

5.2.3.3 template<typename T > void List< T >::push (Element< T > * *elem,* Direction *dir*)

dodaje element na początek/koniec(zależy od implementacji) listy

Parameters

elem	element umieszczany na początku/końcu listy
dir	określa czy włożyć element na początek(Front), na koniec (Back) listy

5.2.3.4 template<typename T > unsigned int List< T >::size ( )

zwraca rozmiar użytej struktury danych

Returns

rozmiar użytej struktury danych

The documentation for this class was generated from the following file:

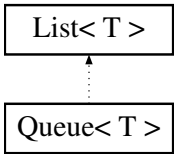
- inc/List.h

5.3 Queue< T > Class Template Reference

Klasa reprezentująca podstawy konterner danych kolejke.

`#include <Queue.h>`

Inheritance diagram for Queue< T >:



Public Member Functions

- Queue ()
  Konstruktor wywołujący konstruktor klasy bazowej List.
- Element< T > * pop ()
  zwraca element z początku kolejki
- void push (Element< T > *elem)
  dodaje element na koniec kolejki
- virtual ~Queue ()
  Wywołuje destruktor klasy bazowej.

5.3.1 Detailed Description

template<class T>class Queue< T >

Klasa reprezentująca podstawy konterner danych kolejke.

Kolejka jest strukturą danych typu typu FIFO, First In, First Out; pierwszy na wejściu, pierwszy na wyjściu

5.3.2 Member Function Documentation

5.3.2.1 template<typename T > Element< T > * Queue< T >::pop ( )

zwraca element z początku kolejki

Returns

element będący na początku kolejki

5.3.2.2 template<typename T > void Queue< T >::push (Element< T > * *elem*)

dodaje element na koniec kolejki

Parameters

elem	element umieszczany na końcu

The documentation for this class was generated from the following file:

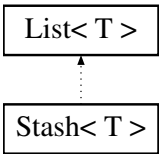
- inc/Queue.h

5.4 Stash< T > Class Template Reference

Klasa reprezentująca podstawy konterner danych - stos.

`#include <Stash.h>`

Inheritance diagram for Stash< T >:



Public Member Functions

- Stash ()

  Konstruktor wywołujący konstruktor klasy bazowej List.
- Element< T > * pop ()

  zwraca element z wierzchu stosu
- void push (Element< T > *elem)

  dodaje element na wierzch
- virtual ~Stash ()

  Wywołuje destruktor klasy bazowej.

5.4.1 Detailed Description

template<class T>class Stash< T >

Klasa reprezentująca podstawy konterner danych - stos.

Kolejka jest strukturą danych typu typu LIFO, Last In, First Out; ostatni na wejściu, pierwszy na wyjściu

5.4.2 Member Function Documentation

5.4.2.1 template<typename T > Element< T > * Stash< T >::pop ( )

zwraca element z wierzchu stosu

Returns

element będący na wierzchu stosu

5.4.2.2 template<typename T > void Stash< T >::push (Element< T > * *elem*)

dodaje element na wierzch

Parameters

elem	element, który zostanie umieszczony na wierzchu stosu

The documentation for this class was generated from the following file:

- inc/Stash.h

5.5 Timer Class Reference

Klasa do pomiaru różnicy czasów.

`#include <Timer.h>`

Public Member Functions

- Timer ()
  Konstruktor zerujący parametry.
- void startTimer ()
  Zmierzenie czasu rozpoczęcia pomiaru.
- void stopTimer ()
  Zmierzenie czasu zakończenia pomiaru.
- double diffTimeMs ()
  Funkcja zwracająca różnice czasu pomiędzy czasem rozpoczęcia i zakończenia pomiaru.

5.5.1 Detailed Description

Klasa do pomiaru różnicy czasów.

Klasa pozwala na pomiar czasów w danych momentach oraz na zwrócenie czasu, który upłynał pomiędzy tymi momentami

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Timer::Timer ( )

Konstruktor zerujący parametry.

Konstruktor ten odpowiada za zerowania zmiennych startu i stopu w celu możliwości późniejszego sprawdzenia, czy pomiary czasu konieczne do wyznaczenia różnicy zostały zrealizowane.

5.5.3 Member Function Documentation

5.5.3.1 double Timer::diffTimeMs ( )

Funkcja zwracająca różnice czasu pomiędzy czasem rozpoczęcia i zakończenia pomiaru.

Różnica czasu zwracana jest w milisekundach.

Precondition

Czas zkończenia pomiaru musi być większy (późniejszy) od czasu jego rozpoczęcia

Returns

Zwracana jest różnica czasu zrzutowana do typu double

5.5.3.2 void Timer::startTimer ( )

Zmierzenie czasu rozpoczęcia pomiaru.

Funkcja zapamiętuje bierzący czas, jako czas rozpoczęcia pomiaru.

5.5.3.3 void Timer::stopTimer ( )

Zmierzenie czasu zakończenia pomiaru.

Funkcja zapamiętuje bierzący czas, jako czas zakończenia pomiaru.

The documentation for this class was generated from the following files:

- inc/Timer.h
- src/Timer.cpp

Chapter 6

File Documentation

6.1 inc/Element.h File Reference

Deklaracja i definicja klasy Element.

```
#include <stddef.h>
```

Classes

- class Element< T >

  Klasa reprezentująca abstrakcyjny "pojemnik na dane".

6.1.1 Detailed Description

Deklaracja i definicja klasy Element. Element.h

6.2 inc/List.h File Reference

Deklaracja klasy Cointainer.

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

Classes

- class List< T >

  Klasa reprezentująca podstawy konterner danych z którego korzystają inne - Listę

6.2.1 Detailed Description

Deklaracja klasy Cointainer. List.h

6.3 inc/Queue.h File Reference

Deklaracja i definicja klasy Queue.

```
#include "List.h"
```

Classes

- class Queue< T >

  Klasa reprezentująca podstawy konterner danych kolejke.

6.3.1 Detailed Description

Deklaracja i definicja klasy Queue. Queue.h

6.4 inc/Stash.h File Reference

Deklaracja i definicja klasy Stash.

```
#include "List.h"
```

Classes

- class Stash< T >

  Klasa reprezentująca podstawy konterner danych - stos.

6.4.1 Detailed Description

Deklaracja i definicja klasy Stash. Stash.h

6.5 inc/Timer.h File Reference

Plik zawierający deklaracje klasy Timer służącej do pomiaru różnicy czasów.

```
#include <ctime>
```

Classes

- class Timer

  Klasa do pomiaru różnicy czasów.

6.5.1 Detailed Description

Plik zawierający deklaracje klasy Timer służącej do pomiaru różnicy czasów. Timer.h

6.6 src/Element.cpp File Reference

Definicja klasy Element.

```
#include "../inc/Element.h"
```

6.6.1 Detailed Description

Definicja klasy Element. Element.cpp

6.7 src/List.cpp File Reference

Definicja klasy List.

```
#include "../inc/List.h"
```

6.7.1 Detailed Description

Definicja klasy List. List.cpp

6.8 src/main.cpp File Reference

Plik zawierający sekwencje operacji do mierzenia czasu operacji mnożenia elementów tablicy przez 2.

```
#include <iostream>
#include "../inc/Timer.h"
#include "../inc/List.h"
#include "../inc/Queue.h"
#include "../inc/Stash.h"
#include "../inc/Element.h"
#include <math.h>
```

Functions

- int main ()

6.8.1 Detailed Description

Plik zawierający sekwencje operacji do mierzenia czasu operacji mnożenia elementów tablicy przez 2. main.cpp

6.8.2 Function Documentation

6.8.2.1 int main ( )

liczba określająca wykładnik liczby elementów, które umieścimy w danej strukturze danych

ilość testowanych wielkości strukur danych

reprezentacja (obiekt) listy przechowujący int'y

reprezentacja (obiekt) kolejki przechowujący int'y

reprezentacja (obiekt) stosu przechowujący int'y

wyniki pomiarów czasu dla listy

wyniki pomiarów czasu dla kolejki

wyniki pomiarów czasu dla stosu

czasomierz użyty do mierzenia czasu wypełniania struktur

6.9 src/Queue.cpp File Reference

Definicja klasy Queue.

```
#include "../inc/Queue.h"
```

6.9.1 Detailed Description

Definicja klasy Queue. Queue.cpp

6.10 src/Stash.cpp File Reference

Definicja klasy Stash.

```
#include "../inc/Stash.h"
```

6.10.1 Detailed Description

Definicja klasy Stash. Stash.cpp

6.11 src/Timer.cpp File Reference

Plik zawierający definicje funkcji klasy Timer służącej do pomiaru różnicy czasów.

```
#include "../inc/Timer.h"
#include <iostream>
```

6.11.1 Detailed Description

Plik zawierający definicje funkcji klasy Timer służącej do pomiaru różnicy czasów. Timer.cpp

Index