Lab4

Wygenerowano przez Doxygen 1.7.6.1

Sun Mar 23 2014 23:35:34

Spis treści

Rozdział 1

Sortowanie

Autor

Krzysztof Kucharczyk

Data

23.03.2014

Wersja

1

Program umożlwia sortowanie trzema metodami:

1) Sortowanie szybkie (ang. Quick Sort)

2) Sortowanie kopcowe (ang. Heap Sort)

3) Sortowanie przez scalanie (ang. Merge Sort)

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

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

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy Pomiar::Chrono

Stworzona do pomiarów czasu.

`#include <chrono.h>`

Metody publiczne

- Chrono ()

  Konstruktor rozpoczynający pomiar czasu.
- clock::time_point restart ()

  Resetuje zegar.
- microseconds elapsedUs ()

  Zwraca czas działania w mikrosekundach.
- milliseconds elapsedMs ()

  Zwraca czas działania w milisekundach.
- nanoseconds elapsedNs ()

  Zwraca czas działania w nanosekundach.
- void Eksportuj_dane (double czas)

  Eksportuje dane do pliku.

4.1.1 Opis szczegółowy

Stworzona do pomiarów czasu.

Klasa służy do mierzenia czasu wykonywania metod. Możliwe jest uzyskanie czasu działania w:

- milisekundach,

- mikrosekundach,
- nanosekundach.

  Dodatkowo klasa umożliwia zapisywanie wyników pracy do plików o nazwie "-Wyniki_temp.txt", które analizowane są przez program do bechmarku.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 Pomiar::Chrono::Chrono ( ) [inline]

Konstruktor rozpoczynający pomiar czasu.

Konstruktor pozwala w prosty sposób rozpocząć pomiar czasu, tj. poprzez zdefiniowanie obiektu klasy Chrono.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 void Pomiar::Chrono::Eksportuj_dane (double *czas*) [inline]

Eksportuje dane do pliku.

Metoda umożliwia zapisanie upływu czasu fragmentu kodu do pliku o nazwie "Wyniki_-temp.txt". Pomiary są dopisywane, dzięki czemu dane nie są tracone.

Parametry

in	*czas*	Czas, który ma zostać zapisany

4.1.3.2 milliseconds Pomiar::Chrono::elapsedMs ( ) [inline]

Zwraca czas działania w milisekundach.

Metoda zwraca czas działania konkretnego fragmentu kodu w milisekundach.

Zwraca

Zwraca czas w milisekundach

4.1.3.3 nanoseconds Pomiar::Chrono::elapsedNs ( ) [inline]

Zwraca czas działania w nanosekundach.

Metoda zwraca czas działania konkretnego fragmentu kodu w nanosekundach.

Zwraca

Zwraca czas w nanosekundach

4.1.3.4 microseconds Pomiar::Chrono::elapsedUs ( ) [inline]

Zwraca czas działania w mikrosekundach.

Metoda zwraca czas działania konkretnego fragmentu kodu w mikrosekundach.

Zwraca

Zwraca czas w mikrosekundach

4.1.3.5 clock::time_point Pomiar::Chrono::restart ( ) [inline]

Resetuje zegar.

Metoda resetuje i załącza ponownie pomiar.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- /home/krzysztof/Desktop/PAMSI/Laboratorium/Laboratorium_4/Sortowanie/inc/chrono.h

4.2 Dokumentacja klasy Obiekt

Klasa umożliwia tworzenie przydatnej struktury.

`#include <obiekt.h>`

Metody publiczne

- Obiekt ()
  Inicjuje obiekt pustym wskaźnikiem.
- Obiekt (int ∗Wskaznik)
  Inicjuje obiekt określonym wskaźnikiem.
- Obiekt operator+ (const Obiekt &Nowy)
  Pozwala dodawać dwa obiekty.
- Obiekt & operator= (const Obiekt &Nowy)
  Pozwala przypisywać jeden obiekt do drugiego.
- bool operator== (const Obiekt &Nowy)
  Pozwala intuicyjnie porównywać obiekty.
- void Show ()
  Umożlwia zwizualizowanie elementów obiektu.
- void Pobierz_dane (string nazwa_pliku)
  Pobiera dane z określonego pliku.
- void Pomnoz (int mnoznik)
  Mnoży elemnty tablicy Obiektu.

- void Czy_rowne (const Obiekt &Porownywany)

  Sprawdza, czy Obiekty są sobie równe.
- void Zamien_elementy (int i, int j)

  Zamienia elementy o podanych indeksach.
- void Odwroc_kolejnosc ()

  Odwraca kolejność tablicy.
- void Dodaj_element (int element)

  Dodaje jeden element do tablicy.
- void Dodaj_elementy (Obiekt &Nowy)

  Dodaje wiele elementów do tablicy.

Atrybuty publiczne

- int * Tablica

  Umożliwia tworzenie uniwersalnych tablicy.

4.2.1 Opis szczegółowy

Klasa umożliwia tworzenie przydatnej struktury.

Klasa pozwala na tworzenie bardzo przydatnej, uniwersalnej struktury bardzo podatnej na wszelki modyfikacje (np. sortowanie). Jej najważniejszym elementem jest wskaźnik umozliwiający stworzenie dynamicznej tablicy. Dane w niej przechowywane mogą być w różnoraki sposób modyfikowane dzięki zaimplementowanym metodom.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Obiekt::Obiekt ( ) [inline]

Inicjuje obiekt pustym wskaźnikiem.

Prosty konstruktor, jego zadanie ogranicza się do dezaktywowania wskaźnika, by w razie czego na nic nie pokazywał i nie wprowadzał w ewentualny błąd.

4.2.2.2 Obiekt::Obiekt (int * *Wskaznik*) [inline]

Inicjuje obiekt określonym wskaźnikiem.

Kontruktor pozwala zainicjować obiekt już istniejącą tablicą danych, dzięki czemu nie trzeba jej ponownie przepisywać.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 void Obiekt::Czy_rowne (const Obiekt & *Porownywany*)

Sprawdza, czy Obiekty są sobie równe.

Metoda sprawdza, czy dwa obiekty są sobie równe. Milczy w przypaku optymistycznym, wyświetla stosowny komunikat w przypadku przeciwnym.

Parametry

in	- *Porownywany*	Nazwa porównywanego obiektu

4.2.3.2 void Obiekt::Dodaj_element (int *element*)

Dodaje jeden element do tablicy.

Metoda dodaje jeden element na koniec tablicy poprzez stworzenie nowej, o jeden większej tablicy, do której przepisywane są dane ze starej i dodawany nowy element na samym końcu. Stara tablica zostaje usunięta. Oczywiście element zerowy zostaje adekwatnie uaktualniony.

Parametry

in	*element*	Nowy element, który zostanie dodany na koniec tablicy

4.2.3.3 void Obiekt::Dodaj_elementy (Obiekt & *Nowy*)

Dodaje wiele elementów do tablicy.

Metoda umożliwia dodanie dużej ilości elementów na koniec istniejącej w obiekcie tablicy. Działa na zasadzie stworzenia nowej, większej tablicy i umieszczeniu w niej elementów obu tablic. Stara tablica zostaje zwolniona, a indeks zerowy uaktualniony.

Parametry

Nowy	Obiekt Obiekt, którego tablica zostaje wpisana na koniec istniejącej tablicy w Obiekcie.

4.2.3.4 void Obiekt::Odwroc_kolejnosc ( )

Odwraca kolejność tablicy.

Metoda powoduje inwersję elementów tablicy. Oczywiście element zerowy zostaje na miejscu.

4.2.3.5 Obiekt Obiekt::operator+ (const Obiekt & *Nowy*)

Pozwala dodawać dwa obiekty.

Przeciążenie to pozwala dodawać tablicę jednego obiektu do końca tablicy drugiego obiektu. Dzięki modyfikatorowi const obiekt dodawany ma zapewnioną nietykalność.

4.2.3.6 Obiekt & Obiekt::operator= (const Obiekt & *Nowy*)

Pozwala przypisywać jeden obiekt do drugiego.

Przeciążenie umożliwia w prosty sposób przypisanie wartości jednego obiektu do drugiego. Obiekt, który jest przepisywany jest nietykalny dzięki modyfikatorowi const.

4.2.3.7 bool Obiekt::operator== (const Obiekt & *Nowy*)

Pozwala intuicyjnie porównywać obiekty.

Przeciążenie umożliwia intuicyjne porównywanie dwóch elementów. Zwracana jest jedna z dwóch wartości:

- 1 - gdy oba obiekty są identyczne,
- 0 - gdy oba obiekty są różne.

4.2.3.8 void Obiekt::Pobierz_dane (string *nazwa_pliku*)

Pobiera dane z określonego pliku.

Metoda umożliwia pobranie danych pliku o określonej nazwie. Plik jest sprawdzany pod względem swojego formatu, tj. czy zachowany jest następujący układ:

```
* indeks
* 1.              | ilość elementów
* 2.              | element_1
* 3.             | element_2
* .
* .
* .
* ilość_elementów+1 | element ilość_elementów-1
*
```

Parametry

in	*nazwa_pliku*	Zawiera nazwę pliku, z którego pobierane są dane

4.2.3.9 void Obiekt::Pomnoz (int *mnoznik*)

Mnoży elemnty tablicy Obiektu.

Metoda służy do multiplikowania elementów tablicy przez określoną liczbę.

Parametry

in	*mnoznik*	Wartość, o jaką mają zostać pomnożone elementy

4.2.3.10 void Obiekt::Show ( )

Umożlwia zwizualizowanie elementów obiektu.

Metoda pozwala wypisać całą zawartość obiektu (z wyjątkiem elementu zerowego, tj. długośvi tablicy).

4.2.3.11 void Obiekt::Zamien_elementy (int *i*, int *j*)

Zamienia elementy o podanych indeksach.

Metoda zamienia miejscami elementy tablicy o podanych indeksach. Jedynym ograniczeniem jest indeks zerowy, który zawiera informację o długości tablicy. Jego zmiana jest niemożliwa, o tym także poinformuje program w razie ewentualnego błędu.

Parametry

in	*i*	Indeks pierwszego elementu
in	*j*	Indeks drugiego elementu

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 int* Obiekt::Tablica

Umożliwia tworzenie uniwersalnych tablicy.

Pole pozwala na tworzenie uniwersalnych tablic o zdefiniowanych przez użytkownika wielkościach. Pole to jest podstawowym polem klasy.

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/krzysztof/Desktop/PAMSI/Laboratorium/Laboratorium_4/Sortowanie/inc/obiekt.h
- /home/krzysztof/Desktop/PAMSI/Laboratorium/Laboratorium_4/Sortowanie/src/obiekt.cpp

4.3 Dokumentacja klasy Sort

Umożliwia sortowanie tablic.

```
#include <sort.h>
```

Metody publiczne

- void Zapisz_dane (const Obiekt Obiekt)

  Przepisuje dane z obiektu Obiekt.
- void Wizualizuj_zmiany (const Obiekt Obiekt)

  Pokazuje zmiany w przystępnej formie.
- void QuickSort (Obiekt &Obiekt, int pierwszy, int ostatni)

  Sortowanie elementów metodą Quicksort.
- void StrukturaKopca (Obiekt &Obiekt, int id, int hs)

  Tworzy strukturę kopca.
- void BudujKopiec (Obiekt &Obiekt, int hs)

  Tworzy kopiec binarny z tablicy danych.
- void HeapSort (Obiekt &Obiekt)

  Sortowanie tablicy metodą kopcową
- void Merge (Obiekt &Obiekt, int poczatek, int koniec, int srodek)

  Dzieli tablicę Obiektu na dwie części.
- void MergeSort (Obiekt &Obiekt, int poczatek, int koniec)

  Sortuje tablicę obiektu za pomocą algorytmu sortowania przez scalanie.

Atrybuty publiczne

- int * Zapisane_dane

  Zapisuje dane z obiektu Obiekt.

4.3.1 Opis szczegółowy

Umożliwia sortowanie tablic.

Klasa służy do sortowania tablic zawartych w obiektach klasy Obiekt. Posiada metody umozliwiające zarówno sortowanie elementów tablic różnymi algorytmami o różnej wydajności, jak i zapisać aktualne dane z obiektu i przedstawić zestawienie danych (przed sortowanie, po sortowaniu).

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 void Sort::BudujKopiec (Obiekt & *Obiekt,* int *hs*)

Tworzy kopiec binarny z tablicy danych.

Metoda pozwala stworzyć kopiec z określonego elementu Obiekt. Inicjuje metodę - StrukturaKopca i za jej pomocą, dla określonych elementów, zmienia tablicę w preposortowany kopiec, którego struktura umożliwia działanie algorytmowi sortowania kopcowego, w wypadku tego programu jest to HeapSort.

Parametry

in	*Obiekt*	Obiekt, którego tablica ma zostać posortowana
in	*hs*	Licznik określający ilość przebiegów sortowania, pochodzi z metody HeapSort

4.3.2.2 void Sort::HeapSort (Obiekt & *Obiekt*)

Sortowanie tablicy metodą kopcową

Metoda umożliwia posortowanie tablicy Obiektu za pomocą algorytmu sortowania kopcowego (ang. HeapSort). Algorytm działa na zasadzie ustawiania największego elementu sortowanej tablicy (korzenia) na pierwszym miejscu posortowanych elementów, następnie jego stare miejsce zostaje zastąpione elementem z najniższej gałęzi. Dzięki metodzie naprawiania, struktura kopca zostaje naprawiona i na szczycie ponownie zostaje umieszczony największy element, który z korzenia wędruje na drugie miejsce posortowanej tablicy, etc. Dzięki temu tablica wypełnia się posortowanymi liczbami w kolejności malejącej (najpierw liczby największe) potem mniejsze.

Parametry

in	*Obiekt*	Obiekt, którego tablica ma zostać posortowana

4.3.2.3 void Sort::Merge (Obiekt & *Obiekt,* int *poczatek,* int *koniec,* int *srodek*)

Dzieli tablicę Obiektu na dwie części.

Metoda jest pomocniczą metodą algorytmu sortowania przez scalanie. Jej rolą jest podział tablicy Obiektu na dwie równe części.

Parametry

in	*Obiekt*	Obiekt, którego tablica ma zostać posortowana
in	*low*	Określa początek tablicy
in	*high*	Określa koniec tablicy
in	*mid*	Określa środek tablicy

4.3.2.4 void Sort::MergeSort (Obiekt & *Obiekt,* int *poczatek,* int *koniec*)

Sortuje tablicę obiektu za pomocą algorytmu sortowania przez scalanie.

Metoda umożliwia posortowanie tablicy obiektu Obiekt poprzez zastosowanie rekurencyjnego algorytmu sortowania przez scalanie (ang. Merge sort). Metoda ta ogranicza się do rekurencyjnego wywoływania siebie i metody Merge. Tablica przeznaczona do sortowania jest dzielona aż do oporu lub jednego pozostałego elementu, a następnie, mając posortowane kawałki, łączy części w jedną, posortowaną, wynikową tablicę.

Parametry

in	*Obiekt*	Obiekt, którego tablica ma zostać posortowana
in	*low*	Określa początek stosowania algorytmu (tablicy)
in	*high*	Określa konec stosowania algorytmu (tablicy)

4.3.2.5 void Sort::QuickSort (Obiekt & *Obiekt,* int *pierwszy,* int *ostatni*)

Sortowanie elementów metodą Quicksort.

Metoda służy do sortowanie elementów tablicy obiektu przesyłanej jako argument metody. Algorytm sortowania szybkiego (ang. quicksort) polega na dzieleniu tablicy na podtablice zgodnie z określonym kryterium (tzw. punkt osiowy), jedna tablica tworzona jest z mniejszych od elementu osiowego elementu, druga z większych od elementu osiowego elemntów. Następnie rekurencyjnie powtarzane jest dzielenie tablic na podtablice aż do tablic jednoelemntowych.

Parametry

Obiekt	Obiekt, którego tablica ma zostać posortowana
pierwszy	Indeks pierwszego elementu, od którego ma się rozpocząć sortowanie
ostatni	Indeks ostatniego elementu, który ogranicza sortowanie

4.3.2.6 void Sort::StrukturaKopca (Obiekt & *Obiekt,* int *id,* int *hs*)

Tworzy strukturę kopca.

Metoda tworzy strukturę kopca binarnego z przesyłanej jako argument tablicy obiektu Obiekt. Dzięki rozdzielaniu części określonej jako id na korzeń i dwójkę potomstwa, możliwe jest szybkie rejestrowanie, czy struktura kopca jest w pełni zachowana. W przypadku dowolnych niezgodności, kopiec zostaje szybko naprawiony.

Parametry

in	*Obiekt*	Obiekt, którego tablica ma zostać posortowana
in	*id*	Numer elementu, który wraz z potomstwem ma być badany
in	*hs*	Licznik określający ilość przebiegów sortowania, pochodzi z metody HeapSort

4.3.2.7 void Sort::Wizualizuj_zmiany (const Obiekt *Obiekt*)

Pokazuje zmiany w przystępnej formie.

Metoda wyświetla dane w postaci dwóch kolumn: lewa odpowiada za dane nie posortowane, prawa za posortowane. Wszystkie elementy są indeksowane. Służy tylko i wyłącznie w celu wizualizacji i nie ma żadnego praktycznego znaczenia w ujęciu algorytmów sortowania. Tablica z której pobierane są dane pozostaje nie naruszona dzięki modyfikatorowi const.

Parametry

Obiekt	Obiekt z którego mają zostać przepisane dane

4.3.2.8 void Sort::Zapisz_dane (const Obiekt *Obiekt*)

Przepisuje dane z obiektu Obiekt.

Zadanie metody jest przepisanie elementów z obiektu Obiekt do wewnrznej tablicy, aby później możliwe było przystępne przedstawienie owych danych i łatwiejsza lokalizacja błędów. Tablica z której pobierane są dane pozostaje nie naruszona dzięki modyfikatorowi const.

Parametry

Obiekt	Obiekt z którego mają zostać przepisane dane

4.3.3 Dokumentacja atrybutów składowych

4.3.3.1 int* Sort::Zapisane_dane

Zapisuje dane z obiektu Obiekt.

Pole służy przy wizualizacji zmian zachodzących w tablicach obiektów. Nie ma żadnego praktycznego znaczenia w ujęciu sortowania elementów.

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/krzysztof/Desktop/PAMSI/Laboratorium/Laboratorium_4/Sortowanie/inc/sort.h
- /home/krzysztof/Desktop/PAMSI/Laboratorium/Laboratorium_4/Sortowanie/src/sort.cpp

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku /home/krzysztof/Desktop/PAMSI/Laboratorium/-Laboratorium_4/Sortowanie/inc/chrono.h

Zawiera definicję klasy Chrono.

`#include <chrono> #include <fstream>`

Komponenty

- class Pomiar::Chrono

  Stworzona do pomiarów czasu.

Definicje typów

- typedef std::chrono::high_resolution_clock **Pomiar::clock**
- typedef std::chrono::microseconds **Pomiar::microseconds**
- typedef std::chrono::milliseconds **Pomiar::milliseconds**
- typedef std::chrono::nanoseconds **Pomiar::nanoseconds**

Funkcje

- clock::time_point **Pomiar::now** ()
- microseconds **Pomiar::intervalUs** (const clock::time_point &t1, const clock-::time_point &t0)
- milliseconds **Pomiar::intervalMs** (const clock::time_point &t1, const clock::time-_point &t0)
- nanoseconds **Pomiar::intervalNs** (const clock::time_point &t1, const clock::time-_point &t0)

5.1.1 Opis szczegółowy

Zawiera definicję klasy Chrono. Plik zawiera definicję klasy Chrono.

5.2 Dokumentacja pliku /home/krzysztof/Desktop/PAMSI/Laboratorium/-Laboratorium_4/Sortowanie/inc/obiekt.h

Zawiera definicję klasy Obiekt.

```
#include <cstdlib> #include <iostream> #include <fstream> ×
#include <string>
```

Komponenty

- class Obiekt

  Klasa umożliwia tworzenie przydatnej struktury.

5.2.1 Opis szczegółowy

Zawiera definicję klasy Obiekt. Plik zawiera definicję klasy Obiekt.

5.3 Dokumentacja pliku /home/krzysztof/Desktop/PAMSI/Laboratorium/-Laboratorium_4/Sortowanie/inc/sort.h

Definicja klasy Sort.

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

Komponenty

- class Sort

  Umożliwia sortowanie tablic.

5.3.1 Opis szczegółowy

Definicja klasy Sort. Plik zawiera definicję klasy Sort.