Laboratorium 1&2

Wygenerowano przez Doxygen 1.7.6.1

Sat Mar 15 2014 01:08:12

Spis treści

Rozdział 1

Prosty program bazowy

Autor

Krzysztof Kucharczyk

Data

10.03.2014

Wersja

6

Program służy do operowania na obiektach klasy Obiekt, które umożliwiają pracę na tablicach. Dzięki wykorzystaniu budowy obiektowej program posiada zwartą strukturę.

1.1 Najwazniejsze cechy

Program działa na zasadzie tworzenia obiektów i zawartych w nich tablic dynamicznych. Prosta implementacja klas i metod pozwala na sprawne i intuicyjne operowanie dostarczonymi zasobami. Dodatkowo program wyposażony jest w moduł umożliwiający komunikację z programem ./benchmark, skupionym na obliczaniu efektywności tworzonych algorytmów.

1.2 Funkcjonalność programu

Program pozwala na tworzenie obiektów oraz operacje pomiaru czasu konkretnych elementów kodu. Do najważniejszych możliwości programu można zaliczyć:

-> pobieranie danych z plików tekstowych o określonym formacie -> wykonywanie operacji na obiektach (w przyszłości sortowanie) -> wyświetlanie danych zawartych w obiektach -> zaimplementowane przeciążenia umożliwiające intuicyjne korzystanie z zasobów

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 Obiekt

Metody publiczne

- Obiekt ()
- Obiekt (int ∗Wskaznik)
- Obiekt operator+ (const Obiekt &Nowy)
- Obiekt & operator= (const Obiekt &Nowy)
- bool operator== (const Obiekt &Nowy)
- void Show ()
- void Pobierz_dane (string nazwa_pliku)
- void Pomnoz (int mnoznik)
- void Czy_rowne (const Obiekt &Porownywany)
- void Zamien_elementy (int i, int j)
- void Odwroc_kolejnosc ()
- void Dodaj_element (int element)
- void Dodaj_elementy (Obiekt &Nowy)

Atrybuty publiczne

- int ∗ **Tablica**

4.1.1 Dokumentacja konstruktora i destruktora

4.1.1.1 Obiekt::Obiekt ( ) [inline]

Podstawowy konstruktor klasy, inicjuje pole Tablica wskaźnikiem NULL.

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

Konstruktor pozwala na zainicjowanie obiektu klasy odpowiednim wskaźnikiem.

Parametry

Wskaznik	- wskaznik typu int wskazujący na kokretną strukturę. Zostaje przypisany do pola Tablica.

4.1.2 Dokumentacja funkcji składowych

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

Metoda analogiczna do operator ==. Pozwala na porównanie, czy dwa obiektu są sobie równe. W przeciwieństwie do wspomnianego przeładowania metoda ta nie reaguje, gdy dane są identyczne, a w przypadku różnych danych wyświetla stosowny komunikat.

Parametry

Porownywany	- obiekt, który podlega porownaniu. Nietykalny dzieki const.

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

Metoda pozwala na dodanie elementu pobranego jako argument do Tablicy. Metoda działa na zasadzie skopiowania zawartości Tablicy do nowej, większej o jeden element tablicy. Ostatnie miejsce nowej tablicy wypełnia nowy element. Długość tablicy zostaje odpowiednio zaktualizowana.

Parametry

element	- element, który ma zostać dodany

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

Metoda umożliwia dodanie dwóch obiektów do siebie w sensie dołączenia jednej Tablicy do drugiej. Metoda za argument przybiera Obiekt, który ma zostać dodany. Długość - Tablicy zostaje odpowiednio zaktualizowany.

Parametry

Nowy	- obiekt, który ma zostać dopisany

4.1.2.4 void Obiekt::Odwroc_kolejnosc ( )

Metoda pozwala odwrócić kolejność elementów w Tablicy.

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

Przeciążenie operatora +. Pozwala na wykonywania intuicyjnych operacji dodawania dwóch obiektów klasy poprzez dołączenie jednej tablicy do drugiej. Tablica dodawana na końcu jest argumentem przeładowania, modyfikator const chroni jego zawartość.

Parametry

Nowy	- obiekt klasy obiekt, który zostaje dodany do obiektu po lewej stronie znaku +. Niezmienność zapewniona const'em.

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

Przeciążenie operatora =. Umożliwia podstawienie parametrów jednego obiektu do drugiego. Obiekt przypisywany ma modyfikator const chroniący jego zawartość.

Parametry

Nowy	- obiekt klasy obiekt, który zostaje przepisany do obiektu po lewej stronie znaku =. Niezmienność zapewniona const'em.

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

Przeciążenie operatora ==. Pozwala na porównywanie zawartości dwóch obiektów i zwrócenie wyniku porównywania w postaci bool (prawda/fałsz). Obiekt porównywany ma modyfikator const chroniący jego zawartość.

Parametry

Nowy	- obiekt klasy obiekt, który zostaje porównany do obiektu po lewej stronie operatora ==. Const zapewnia nietykalność argumentu.

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

Metoda umożliwiająca wypełnienie Tablicy obiektu elementami z pliku o nazwie podanej jako argument. Metoda sprawdza poprawność pliku, tj. czy format pliku mieści się w następującym:

```
1   | n - ilość wierszy
2   | dane(1)
3   | dane(2)
.   | .
```

```
.   | .
.   | .
n+1 | dane(n-1)

, gdzie pierwsza kolumna to numeracja wierszy pliku, druga to zadane wartości.
```

Dodatkowo metoda sprawdza czy otworzono plik i czy dane mają logiczny sens.

Parametry

nazwa_pliku	- nazwa pliku, z którego powinny zostać pobrane dane.

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

Metoda mnoży wszystkie elementy Tablicy przez wartość podaną jako argument.

Parametry

mnoznik	- wartość, przez jaką mają być pomnożone wszystkie elementy Tablicy.

4.1.2.10 void Obiekt::Show ( )

Metoda pomocnicza umożliwiająca wyświetlenie zawartości Tablicy konkretnego obiektu.

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

Metoda ta umozliwia zamiane miejscami dwoch elementow Tablicy. Jako argumenty przyjmuje indeksy dwóch elementów. Warto również zaznaczyć, że argumenty muszą być z zakresu

```
1 - n

-> n - ilość elementów
```

ze względu na konkretną budowę tablic.

Parametry

i	- indeks pierwszego elementu do zamiany
j	- indeks drugiego elementu do wymiany

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/krzysztof/Desktop/Workspace/Mnozenie_tablic/inc/obiekt.h
- /home/krzysztof/Desktop/Workspace/Mnozenie_tablic/src/obiekt.cpp

4.2 Dokumentacja klasy Stoper

Metody publiczne

- void Start ()
- void Stop ()
- long int Pokaz_czas_s ()
- long int Pokaz_czas_ns ()
- void Eksport_wyniki ()

Atrybuty publiczne

- timespec **tS**

4.2.1 Dokumentacja funkcji składowych

4.2.1.1 void Stoper::Eksport_wyniki ( ) [inline]

Metoda umożliwia przesyłanie i wymianę danych z innymi programami. Celem stworzenia tej metody jest komunikacja z zewnętrznym programem ./benchmark służącym do oceniania efektywności algorytmu. Metoda tworzy zmienną plikową, która otwiera (lub tworzy, jeżeli nie istnieje) plik o nazwie "Wyniki_temp.txt". Następnie zapisuje w pliku otrzymany w wyniku działania funkcji 'clock_gettime' i 'clock_settime' czas działania algorytmu w nanosekundach [ns].

Metoda sprawdza poprawność otwarcia pliku. Dodatkowo w celu ochrony zapisanych wcześniej danych metoda jedynie dopisuje dane do pliku, nigdy ich nie kasuje.

4.2.1.2 long int Stoper::Pokaz_czas_ns ( ) [inline]

Metoda pomocnicza, służy do zwracania momentu czasu przechowywanego w strukturze tS w nanosekundach.

Parametry

tS.tv_sec	- zapisany moment w nanosekundach

Zwraca

- odpowiedni moment w postaci liczby całkowitej

4.2.1.3 long int Stoper::Pokaz_czas_s ( ) [inline]

Metoda pomocnicza, służy do zwracania momentu czasu przechowywanego w strukturze tS w sekundach.

Parametry

tS.tv_sec	- zapisany moment w sekundach

Zwraca

- odpowiedni moment w postaci liczby całkowitej

4.2.1.4 void Stoper::Start ( ) [inline]

Metoda służy do rozpoczęcia pracy stopera. Jej mechanizm polega na oznaczeniu w przekazywanej zmiennej aktualnego momentu czasu. Metoda przyjmuje jako argument strukturę timespec wyspecjalizowanej do obsługi funkcji związanych z czasem. Operator 'inline' został użyty w celu możliwie najdokładnejszego zachowania momentu wywołania.

Parametry

&tS	- referencja na obiekt struktury timespec, zostaje w nim oznaczony aktualny moment.

4.2.1.5 void Stoper::Stop ( ) [inline]

Metoda służy do zakończenia pracy stopera. Jej mechanizm polega na oznaczeniu w przekazywanej zmiennej aktualnego momentu czasu. Metoda przyjmuje jako argument strukturę timespec wyspecjalizowanej do obsługi funkcji związanych z czasem. Operator 'inline' został użyty w celu możliwie najdokładnejszego zachowania momentu wywołania.

Parametry

&tS	- referencja na obiekt struktury timespec, zostaje w nim oznaczony aktualny moment.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- /home/krzysztof/Desktop/Workspace/Mnozenie_tablic/inc/czas.h

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku /home/krzysztof/Desktop/Workspace/-Mnozenie_tablic/inc/czas.h

Klasa Stoper Klasa ta umożliwia zdobywanie informacji o czasie działania konkretnego elementu programu, w przypadku zajęć PAMSI jest to długość wykonywania poszczególnych operacji. Klasa ta naturalnie współpracuje z programem ./benchmark, gdyż wyeksportowane przez klasę Stoper dane są używane przez program ./benchmark do oceniania długości operacji.

`#include <time.h> #include <fstream> #include <iostream> ×`

Komponenty

- class Stoper

5.1.1 Opis szczegółowy

Klasa Stoper Klasa ta umożliwia zdobywanie informacji o czasie działania konkretnego elementu programu, w przypadku zajęć PAMSI jest to długość wykonywania poszczególnych operacji. Klasa ta naturalnie współpracuje z programem ./benchmark, gdyż wyeksportowane przez klasę Stoper dane są używane przez program ./benchmark do oceniania długości operacji. Klasa pozwala na obliczanie czasu w dwóch jednostkach: -> sekundach -> nanosekundach

Klasa posiada metody umożliwiające: -> Uruchomienie stopera -> Zakończenie działania stopera -> Pokazanie czasu w odpowiednich jednostkach -> Wyeksportowanie długości czasu działania algorytmu do pliku "Wynik_temp.txt", który naturalnie współpracuje z programem ./benchmark.