Sortowania

Generated by Doxygen 1.8.1.2

Mon Mar 31 2014 02:38:10

Contents

1 Program wyliczajacy czas wykonywanego alorytmu

Author

Anna Plywaczyk, 200340

Date

02.03.2014

Version

0,1

Aplikacja porozumiewa sie z uzytkowniem. Prosi o podanie nazwy pliku, na ktorym ma byc wykonany algorytm. W programie zostanie wlaczony stoper, ktorym zmierzymy czas w jakim algorytm zostanie wykonany. Pomiar czasu zostanie wykonany kilkakrotnie i na ekran zostanie wypisana wartosc srednia. Aplikacja ponownie poprosi o podanie nazwy pliku, ktory ma zostac porownany z plikiem pomnozonym przez 2. Jezeli wektory beda jednakowe program zworoci komunikat o tym iz mnozenie zostalo wykonane poprawnie w przeciwnym wypadku zostanie zwrocony komunikat o blednym wykonaniu algorytmu.

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 dane Class Reference

Modeluje pojecie danych, uytych w programie, kt moim przypadku sa wektorem, ktory zostaje wczytany z pliku. Pierwsza zmienna wczytana z pliku jest liczba elemntow wystepujacych w tym pliku. Przyjelam, ze zmienne wczytane z pliku sa liczbami calkowitymi (int).

`#include <dane.hh>`

Public Member Functions

- void **wczytaj** (string nazwa)

  Funkcja wczytujca dane do wektora z pliku. Funkcja otwiera plik zdefiniowany w glownej funkcji przez uytkownika, sprawdza czy plik zosta otwarty, jeeli zosta otwarty wczytana jest liczba elementow pliku, nastepnie wczytywane sa wszystkie liczby do wektora.
- void **wypisz** ()

  Funkcja wypisujca na ekran wszystkie elementy wektora. Funkcja wypisuje na ekran wsyztskie elemnty z pliku na ekran w postaci wektora w kolumnie.
- void **zamien_element** (int i, int j)

  Zamiana elementow Funkcja zamiania dwa elementy wektora, zadane poprzez wywolanie argumentow funkcji.
- void **dodaj_element** (int e)

  Dodawanie elementu. Funkcja dodaje element na koniec wektora. Funkcja zdefiniowana jest poprzez wywolanie argumentow funkcji.
- void **odwroc_element** ()

  Odwracanie elementow Funkcja ogawraca wektor, ostatni element wektora staje sierwszy, a pierwszy ostatnim.
- void **dodaj_elementy** (**dane** wektor2)

  Dodawanie elementu. Funkcja dodaje element na koniec wektora. Funkcja zdefiniowana jest poprzez wywolanie argumentow funkcji.
- int & **operator[]** (int indeks)

  Uzycie operatora [] Przeciazenie operatora stwoarzone abysmy mogli odwolac sie do konkretnego elementu wektora.
- unsigned int **size** ()

  Rozmiar wektora.
- **dane** & **operator+** (**dane** wektor2)

  Uzycie operatora + na wektorze Przeciazenie operatora dodwania, ktory mozemy wykonywac na wektorach.
- **dane** & **operator=** (**dane** wektor2)

  Uzycie operatora = na wektorze Przeciazenie opertora przypisywania.
- vector< int > & **wnetrze** ()

  Metoda dajca dostep do zawartosci wektora danych.
- bool **operator==** (**dane** wektor2)

Operator porownania dwoch wektorow Funkcja, ktora jest operatorem porownania dwoch wektorow.

- void **usun** ()

  Funkcja usuwajca pierwszy elemnt z wektora. Funckja usuwa pierwszy element znajdujcy si wektorze jednoczesnie zmniejszajac liczbe elemntow, funkcja potrzebna przy sortowaniu przez scalanie.

- void **usun_ostatni** ()

  Funkcja usuwajaca ostatni element wektora Funkcja usuwa ostatni element wektora jednoczesnie zmniejszajac liczbe elementow, czyli rozmiar tablicy.

- void **czysc** ()

  Zwalnia pamiec Funkcja usuwa caly wektor z pamieci.

4.1.1 Detailed Description

Modeluje pojecie danych, uytych w programie, kt moim przypadku sa wektorem, ktory zostaje wczytany z pliku. Pierwsza zmienna wczytana z pliku jest liczba elemntow wystepujacych w tym pliku. Przyjelam, ze zmienne wczytane z pliku sa liczbami calkowitymi (int).

Definition at line 31 of file dane.hh.

4.1.2 Member Function Documentation

4.1.2.1 void dane::czysc ( ) [inline]

Zwalnia pamiec Funkcja usuwa caly wektor z pamieci.

Definition at line 158 of file dane.hh.

4.1.2.2 void dane::dodaj_element (int *e*)

Dodawanie elementu. Funkcja dodaje element na koniec wektora. Funkcja zdefiniowana jest poprzez wywolanie argumentow funkcji.

Parameters

in	*e*	- liczba, ktora ma zostac dodana na koniec wektora. return (brak)

Definition at line 49 of file dane.cpp.

4.1.2.3 void dane::dodaj_elementy (dane *wektor2*)

Dodawanie elementu. Funkcja dodaje element na koniec wektora. Funkcja zdefiniowana jest poprzez wywolanie argumentow funkcji.

Parameters

in	*wektor2*	- wektor, ktory ma zostac dodany na koniec wektora, z dwoch zostaje stworzony jeden. return (brak)

Definition at line 66 of file dane.cpp.

4.1.2.4 void dane::odwroc_element ( )

Odwracanie elementow Funkcja ogawraca wektor, ostatni element wektora staje sierwszy, a pierwszy ostatnim.

Returns

(brak)

Definition at line 55 of file dane.cpp.

4.1.2.5 dane & dane::operator+ (dane *wektor2*)

Uzycie operatora + na wektorze Przeciazenie operatora dodwania, ktory mozemy wykonywac na wektorach.

Parameters

in	*wektor2*	- wektor danych ktory ma zostac dodany do wektora glownego

Returns

Zwraca wektor, ktory jest suma dwoch innych

Definition at line 79 of file dane.cpp.

4.1.2.6 dane & dane::operator= (dane *wektor2*)

Uzycie operatora = na wektorze Przeciazenie opertora przypisywania.

Parameters

in	*wektor2*	- wektor danych ktory ma zostac przypisany do wektora glownego

Returns

Zwraca wektor, do ktorego zostal przypisany inny wektor

Definition at line 85 of file dane.cpp.

4.1.2.7 bool dane::operator== (dane *wektor2*)

Operator porownania dwoch wektorow Funkcja, ktora jest operatorem porownania dwoch wektorow.

Parameters

in	*wektor2*	- wektor danych ktory zostaje porownany z danymi glownymi

Returns

True gdy wektory danych sa jednakowe, natomiast jesli nawet jeden element sie rozni od wektora porownywanego zwraca false.

Definition at line 93 of file dane.cpp.

4.1.2.8 int & dane::operator[] (int *indeks*)

Uzycie operatora [] Przeciazenie operatora stwoarzone abysmy mogli odwolac sie do konkretnego elementu wektora.

Parameters

in	*indeks*	- zmienna calkowita, poperzez ktora mozemy dostac do konkretnego elementu wektora.

Returns

Zwraca wartosc jaka znajduje siadanym elemencie wektora.

Definition at line 74 of file dane.cpp.

4.1.2.9 unsigned int dane::size ( ) [inline]

Rozmiar wektora.

Returns

Funkcja zwaraca liczbe elementow wektora.

Definition at line 102 of file dane.hh.

4.1.2.10 void dane::usun ( ) [inline]

Funkcja usuwajca pierwszy elemnt z wektora. Funckja usuwa pierwszy element znajdujcy si wektorze jednoczesnie zmniejszajac liczbe elemntow, funkcja potrzebna przy sortowaniu przez scalanie.

Definition at line 139 of file dane.hh.

4.1.2.11 void dane::usun_ostatni ( ) [inline]

Funkcja usuwajaca ostatni element wektora Funkcja usuwa ostatni element wektora jednoczesnie zmniejszajac liczbe elementow, czyli rozmiar tablicy.

Definition at line 149 of file dane.hh.

4.1.2.12 void dane::wczytaj (string *nazwa*)

Funkcja wczytujca dane do wektora z pliku. Funkcja otwiera plik zdefiniowany w glownej funkcji przez uytkownika, sprawdza czy plik zosta otwarty, jeeli zosta otwarty wczytana jest liczba elementow pliku, nastepnie wczytywane sa wszystkie liczby do wektora.

Parameters

in	*nazwa*	- zmienna, ktostaje wprowadzona przez uytkownika do programu

Returns

(brak)

Definition at line 12 of file dane.cpp.

4.1.2.13 vector<int>& dane::wnetrze ( ) [inline]

Metoda dajca dostep do zawartosci wektora danych.

Returns

Wektor danych.

Definition at line 123 of file dane.hh.

4.1.2.14 void dane::wypisz ( )

Funkcja wypisujca na ekran wszystkie elementy wektora. Funkcja wypisuje na ekran wsyztskie elemnty z pliku na ekran w postaci wektora w kolumnie.

Returns

(brak)

Definition at line 29 of file dane.cpp.

4.1.2.15 void dane::zamien_element (int *i,* int *j*)

Zamiana elementow Funkcja zamiania dwa elementy wektora, zadane poprzez wywolanie argumentow funkcji.

Parameters

in	*i*	- pierwszy numer elementu, ktory ma zostac zamieniony.
in	*j-*	drugi numer elementu, ktory ma zostac zamieniony.

Returns

(brak)

Definition at line 37 of file dane.cpp.

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

- C:/Users/Ania/workspace/zadanie/inc/**dane.hh**
- C:/Users/Ania/workspace/zadanie/src/**dane.cpp**

4.2 kolejkalista Class Reference

Klasa modeluje pojecie kolejki, bazujacej na liscie. Wykonywane sa operacje dodawnia fo kolejki, zdjecia z kolejki, sprawdzenia czy jest pusta oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny.

`#include <kolejka_lista.hh>`

Public Member Functions

- void **enqueue** (int el_dodawany)

  Dodanie elementu z kolejki Funkcja dodaje element na poczatek kolejki.
- void **dequeue** (int ∗a)

  Zdejmowanie elementu z kolejki. Funkcja zdejmuje element z konca kolejki.
- bool **isempty** ()

  Sprawdzanie pojemnosci w kolejce. Funkcja sprawdza czy jest pusta poprzez porownanie liczby_elementow do 0.
- int **size** ()

  Zrowcenie rozmiaru funkcji. Funkcja sprawdza ile elementow jest w kolejce.

4.2.1 Detailed Description

Klasa modeluje pojecie kolejki, bazujacej na liscie. Wykonywane sa operacje dodawnia fo kolejki, zdjecia z kolejki, sprawdzenia czy jest pusta oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny.

Definition at line 29 of file kolejka_lista.hh.

4.2.2 Member Function Documentation

4.2.2.1 void kolejkalista::dequeue (int ∗ *a*)

Zdejmowanie elementu z kolejki. Funkcja zdejmuje element z konca kolejki.

Parameters

in	*a*	- wskaznik do ktorego przypisywany jest element zdejmowany, aby mogl zostal uzyty w przyszlosci

Returns

(brak)

Definition at line 18 of file kolejka_lista.cpp.

4.2.2.2 void kolejkalista::enqueue (int *el_dodawany*)

Dodanie elementu z kolejki Funkcja dodaje element na poczatek kolejki.

Parameters

in	*el_dodawany*	- zmienna stala, ktora ma zostac dodana poczatek kolejki

Returns

(brak)

Definition at line 14 of file kolejka_lista.cpp.

4.2.2.3 bool kolejkalista::isempty ( )

Sprawdzanie pojemnosci w kolejce. Funkcja sprawdza czy jest pusta poprzez porownanie liczby_elementow do 0.

Returns

true jezeli funkcja jest pusta, w przeciwnym wypadku false.

Definition at line 30 of file kolejka_lista.cpp.

4.2.2.4 int kolejkalista::size ( )

Zrowcenie rozmiaru funkcji. Funkcja sprawdza ile elementow jest w kolejce.

Returns

Zwraca liczbe elementow.

Definition at line 34 of file kolejka_lista.cpp.

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

- C:/Users/Ania/workspace/zadanie/inc/**kolejka_lista.hh**
- C:/Users/Ania/workspace/zadanie/src/**kolejka_lista.cpp**

4.3 kolejkatab Class Reference

Klasa modeluje pojecie kolejki, bazujacej na tablicy. Wykonywane sa operacje dodawnia do kolejki, zdjecia z kolejki, sprawdzenia czy jest pusta oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny. Dodatkowo uyte niektore funkcje pomocnicze jak np. powiekszenie tablicy.

`#include <kolejka.hh>`

Public Member Functions

- **kolejkatab** ()

  Konstruktor bezargumentowy. Przy tworzeniu tablica jest pusta, w konstruktorze przydzielana jest dynamicznie, na poczatku tablica ma mozliwosc przypisania tablicy o rozmiarze 1. Jezeli nie uda sie utworzyc tablicy wyrzucany jest blad. Przy wywoaniu konstrktora bezparametrycznego powiekszajaca tablica bedzie dwa razy wieksza.

- void **enqueue** (int el_dodawany)

  Dodanie elementu z kolejki Funkcja dodaje element na poczatek kolejki, jezeli brakuje miejsca podwaja pamiec.
- void **dequeue** (int ∗a)

  Zdejmowanie elementu z kolejki. Funkcja zdejmuje element z konca kolejki, jezeli liczba elementow bedzie mniejsza badz rowna 1/4 mozliwych elementow w tablicy, zostaje ona pomniejszana.
- bool **isempty** ()

  Sprawdzanie pojemnosci w kolejce. Funkcja sprawdza czy jest pusta poprzez porownanie liczby_elementow do 0.
- int **size** ()

  Sprawdzenie rozmiaru funkcji Funkcja sprawdza ile elementow jest w kolejce.

4.3.1 Detailed Description

Klasa modeluje pojecie kolejki, bazujacej na tablicy. Wykonywane sa operacje dodawnia do kolejki, zdjecia z kolejki, sprawdzenia czy jest pusta oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny. Dodatkowo uyte niektore funkcje pomocnicze jak np. powiekszenie tablicy.

Definition at line 29 of file kolejka.hh.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 kolejkatab::kolejkatab ( ) [inline]

Konstruktor bezargumentowy. Przy tworzeniu tablica jest pusta, w konstruktorze przydzielana jest dynamicznie, na poczatku tablica ma mozliwosc przypisania tablicy o rozmiarze 1. Jezeli nie uda sie utworzyc tablicy wyrzucany jest blad. Przy wywoaniu konstrktora bezparametrycznego powiekszajaca tablica bedzie dwa razy wieksza.

Definition at line 79 of file kolejka.hh.

4.3.3 Member Function Documentation

4.3.3.1 void kolejkatab::dequeue (int ∗ *a*)

Zdejmowanie elementu z kolejki. Funkcja zdejmuje element z konca kolejki, jezeli liczba elementow bedzie mniejsza badz rowna 1/4 mozliwych elementow w tablicy, zostaje ona pomniejszana.

Parameters

in	*a*	- wskaznik do ktorego przypisywany jest element zdejmowany, aby mogl zostal uzyty w przyszlosci

Returns

(brak)

Definition at line 58 of file kolejka.cpp.

4.3.3.2 void kolejkatab::enqueue (int *el_dodawany*)

Dodanie elementu z kolejki Funkcja dodaje element na poczatek kolejki, jezeli brakuje miejsca podwaja pamiec.

Parameters

in	*el_dodawany*	- zmienna stala, ktora ma zostac dodana na koniec stosu

Returns

(brak)

Definition at line 39 of file kolejka.cpp.

4.3.3.3 bool kolejkatab::isempty ( )

Sprawdzanie pojemnosci w kolejce. Funkcja sprawdza czy jest pusta poprzez porownanie liczby_elementow do 0.

Returns

true jezeli funkcja jest pusta, w przeciwnym wypadku false.

Definition at line 73 of file kolejka.cpp.

4.3.3.4 int kolejkatab::size ( )

Sprawdzenie rozmiaru funkcji Funkcja sprawdza ile elementow jest w kolejce.

Returns

(brak)

Definition at line 77 of file kolejka.cpp.

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

- C:/Users/Ania/workspace/zadanie/inc/**kolejka.hh**
- C:/Users/Ania/workspace/zadanie/src/**kolejka.cpp**

4.4 stos Class Reference

Klasa modeluje pojecie stosu,bazujacego na tablicy. Na stosie wyonywane sa operacje dodatnia na stos, zdjecia ze stosu, sprawdzenia czy jest pusty oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny. Dodatkowo uyte niektore funkcje pomocnicze jak np. powiekszenie stosu.

```
#include <stos.hh>
```

Public Member Functions

- **stos** ()

  Konstruktor bezargumentowy. Przy tworzeniu tablica jest pusta, w konstruktorze przydzielana jest dynamicznie, na poczatku tablica ma mozliwosc przypisania tablicy o rozmiarze 1. Jezeli nie uda sie utworzyc tablicy wyrzucany jest blad. Przy wywoaniu konstrktora bezparametrycznego powiekszajaca tablica bedzie dwa razy wieksza.
- **stos** (int cos)

  Konstruktor parametryczny. Przy tworzeniu tablica jest pusta, w konstruktorze przydzielana jest dynamicznie, na poczatku tablica ma mozliwosc przypisania tablicy o rozmiarze 1. Jezeli nie uda sie utworzyc tablicy wyrzucany jest blad. Przy wywoaniu konstrktora parametrycznego tablica zostanie powiekszona o 1 element.
- void **push** (int el_dodawany)

  Dodanie elementu na stos. Funkcja dodaje element na koniec stosu, jezeli brakuje miejsca powieksza pamiec.
- void **pop** (int ∗a)

  Zdejmowanie elementu ze stosu. Funkcja zdejmuje element z konca stosu, jezeli liczba elementow bedzie mniejsza badz rowna 1/4 mozliwych elementow w tablicy, zostaje ona pomniejszana.
- bool **isempty** ()

  Zwrocenie rozmiaru stosu Funkcja sprawdza czy stos jest pusty poprzez porownanie liczby_elementow do 0.
- int **size** ()

  Sprawdzenie rozmiaru funkcji Funkcja sprawdza ile elementow jest na stosie.

4.4.1 Detailed Description

Klasa modeluje pojecie stosu,bazujacego na tablicy. Na stosie wyonywane sa operacje dodatnia na stos, zdjecia ze stosu, sprawdzenia czy jest pusty oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny. Dodatkowo uyte niektore funkcje pomocnicze jak np. powiekszenie stosu.

Definition at line 29 of file stos.hh.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 stos::stos ( ) [inline]

Konstruktor bezargumentowy. Przy tworzeniu tablica jest pusta, w konstruktorze przydzielana jest dynamicznie, na poczatku tablica ma mozliwosc przypisania tablicy o rozmiarze 1. Jezeli nie uda sie utworzyc tablicy wyrzucany jest blad. Przy wywoaniu konstrktora bezparametrycznego powiekszajaca tablica bedzie dwa razy wieksza.

Definition at line 79 of file stos.hh.

4.4.2.2 stos::stos (int *cos*) [inline]

Konstruktor parametryczny. Przy tworzeniu tablica jest pusta, w konstruktorze przydzielana jest dynamicznie, na poczatku tablica ma mozliwosc przypisania tablicy o rozmiarze 1. Jezeli nie uda sie utworzyc tablicy wyrzucany jest blad. Przy wywoaniu konstrktora parametrycznego tablica zostanie powiekszona o 1 element.

Definition at line 96 of file stos.hh.

4.4.3 Member Function Documentation

4.4.3.1 bool stos::isempty ( )

Zwrocenie rozmiaru stosu Funkcja sprawdza czy stos jest pusty poprzez porownanie liczby_elementow do 0.

Returns

true jezeli funkcja jest pusta, w przeciwnym wypadku false.

Definition at line 65 of file stos.cpp.

4.4.3.2 void stos::pop (int * *a*)

Zdejmowanie elementu ze stosu. Funkcja zdejmuje element z konca stosu, jezeli liczba elementow bedzie mniejsza badz rowna 1/4 mozliwych elementow w tablicy, zostaje ona pomniejszana.

Parameters

in	*a*	- wskaznik do ktorego przypisywany jest element zdejmowany, aby mogl zostal uzyty w przyszlosci

Returns

(brak)

Definition at line 50 of file stos.cpp.

4.4.3.3 void stos::push (int *el_dodawany*)

Dodanie elementu na stos. Funkcja dodaje element na koniec stosu, jezeli brakuje miejsca powieksza pamiec.

Parameters

in	*el_dodawany*	- zmienna stala, ktora ma zostac dodana na koniec stosu

Returns

(brak)

Definition at line 41 of file stos.cpp.

4.4.3.4 int stos::size ( )

Sprawdzenie rozmiaru funkcji Funkcja sprawdza ile elementow jest na stosie.

Returns

Zwraca liczbe elementow znajdujacych sie na stosie

Definition at line 69 of file stos.cpp.

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

- C:/Users/Ania/workspace/zadanie/inc/**stos.hh**
- C:/Users/Ania/workspace/zadanie/src/**stos.cpp**

4.5 stoslista Class Reference

Klasa modeluje pojecie stosu,bazujacego na liscie. Na stosie wyonywane sa operacje dodatnia na stos, zdjecia ze stosu, sprawdzenia czy jest pusty oraz sprawdzenia rozmiaru.

`#include <stos_lista.hh>`

Public Member Functions

- void **push** (int el_dodawany)

  Dodanie elementu na stos. Funkcja dodaje element na koniec stosu.
- void **pop** (int ∗a)

  Zdejmowanie elementu ze stosu. Funkcja zdejmuje element z konca stosu.
- bool **isempty** ()

  Sprawdzenie czy funkcja jest pusta Funkcja sprawdza czy stos jest pusty poprzez uzycie funkcji size ktora dziala na liscie.
- int **size** ()

  Zwrocenie rozmiaru funkcji Funkcja sprawdza ile elementow jest na stosie.

4.5.1 Detailed Description

Klasa modeluje pojecie stosu,bazujacego na liscie. Na stosie wyonywane sa operacje dodatnia na stos, zdjecia ze stosu, sprawdzenia czy jest pusty oraz sprawdzenia rozmiaru.

Definition at line 25 of file stos_lista.hh.

4.5.2 Member Function Documentation

4.5.2.1 bool stoslista::isempty ( )

Sprawdzenie czy funkcja jest pusta Funkcja sprawdza czy stos jest pusty poprzez uzycie funkcji size ktora dziala na liscie.

Returns

true jezeli funkcja jest pusta, w przeciwnym wypadku false.

Definition at line 29 of file stos_lista.cpp.

4.5.2.2 void stoslista::pop (int * *a*)

Zdejmowanie elementu ze stosu. Funkcja zdejmuje element z konca stosu.

Parameters

in	*a*	- wskaznik do ktorego przypisywany jest element zdejmowany, aby mogl zostal uzyty w przyszlosci

Returns

(brak)

Definition at line 17 of file stos_lista.cpp.

4.5.2.3 void stoslista::push (int *el_dodawany*)

Dodanie elementu na stos. Funkcja dodaje element na koniec stosu.

Parameters

in	*el_dodawany*	- zmienna stala, ktora ma zostac dodana na koniec stosu

Returns

(brak)

Definition at line 12 of file stos_lista.cpp.

4.5.2.4 int stoslista::size ( )

Zwrocenie rozmiaru funkcji Funkcja sprawdza ile elementow jest na stosie.

Returns

Zwraca liczbe elementow znajdujacych sie na stosie.

Definition at line 33 of file stos_lista.cpp.

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

- C:/Users/Ania/workspace/zadanie/inc/**stos_lista.hh**
- C:/Users/Ania/workspace/zadanie/src/**stos_lista.cpp**

4.6 zegar Class Reference

Klasa modeluje uruchomienia gownych wlasciowosci programu. Atrybutem klasy sa stowrzone dwa elemnty klasy dane, na ktorych wykonywane sa dzialania.

`#include <uruchom.hh>`

Public Member Functions

- **zegar** ()

- void **wczytaj_dane_pod** (string nazwa_pliku_pod)

  Wczytanie danych podstawowych Funkcja wczytuje dane, na ktorych wykonywany jest algorytm. Dane te sa glowna funkcja programu.
- void **wczytaj_dane_spr** (string nazwa_pliku_spr)

  Wczytanie danych sprawdzajacych Funkcja wczytuje dane, na ktore zostaja porwonane z danymi na ktorych zostal wykonany algorytm.
- void **algorytm** ()

  Funkcja wykonujca zadany algorytm na wektorze. Funkcja wykonuje zadany algorytm na wektorze wejsciowym. W naszym przypadku wektor pomnoony jest przez sta liczb
- bool **porownaj** ()

  Funkcja porjca dwa wektory. Funkcja porje dwa wektory, sprawdza czy wszystkie elemnty s ze sob r.
- LARGE_INTEGER **wlaczStoper** ()

  Funkcja zapamietujca czas poczatkowy. Funkcja naleca do biblioteki "windows.h", stoper zostaje wczony. Funkcja naleca do funkcji bool QueryPerformanceCounter(_out LARGE_INTEGER ∗lpPerformanceCount), funkcja ta zwraca wartosc niezerowa jeeli waczenie zako sikcesem, natomiast w przeciwnym wypadku zostanie wyrzucony bd i zwrartosc 0. Dla komputerltiprocesorowych nie ma znaczenia, ktest uywany, mog jedynie rnimalnie czasy.
- LARGE_INTEGER **wylaczStoper** ()

  Funkcja zapamietujca czas koy. Funkcja naleca do biblioteki "windows.h", stoper zostaje wyczony, aby zosta zmierzony czas wykonania algorytmu w programie, poprzez odje czasu pocztkowego od czasu koego. Funkcja naleca do funkcji bool QueryPerformanceCounter(_out LARGE_INTEGER ∗lpPerformanceCount), funkcja ta zwraca wartosc niezerowa jeeli wyaczenie zako sikcesem, natomiast w przeciwnym wypadku zostanie wyrzucony bd i zwrartosc 0. Dla komputerltiprocesorowych nie ma znaczenia, ktest uywany, mog jedynie rnimalnie czasy.
- void **wczytaj** (string nazwa)

  Funkcja wczytujca dane do stosu. Funkcja otwiera plik zdefiniowany w glownej funkcji przez uytkownika, sprawdza czy plik zosta otwarty, jeeli zosta otwarty wczytana jest liczba elementow pliku, nastepnie wczytywane sa wszystkie liczby do tablicy.
- **dane scal_sort** (**dane** tab)

  Sortowanie przez scalanie glowne wywolanie funkcji Funkcja sortujaca liczby poprzez scalnie. Ta czesc funkcji dzieli ciag na dwa rowne podciagi, nastepnie kazda z czesci sortujemy, wywolujac rekurencyjnie ta funkcje. Gdy podciagi beda uporzadkowane scalilmy te male podciagi i powstanie nam ciag liczb posortowanych.
- **dane scal** (**dane** lewo, **dane** prawo)

  Funkcja wykorzystana do sortowania przez scalanie Funkcja ta odpowiedzialana jest za scalanie dwoch wektorow w jeden. Porownuje dwa elementy i wiekszy elemnt dodaje do wektora.
- **dane heap** (**dane** tab)

  Funkcja tworzaca kopiec Funkcja ta uzyta jest do sortowania poprzez kopcowanie. Tworzy drzewo binarne czyli kopiec Najwiekszy element jest na szczycie kopca i jest to tzw ojciec, ktory ma dwoch synow ktore sa mniejsze od niego, jezeli nie zamienia miejscami ojca z synem.Tak stowrzony kopiec uporzadkowany jest prawie malejaco.
- **dane heap_sort** (**dane** tab)

  Glowna funkcja sortujaca poprzez kopcowanie Funkcja sortujaca liczby poprzez kopcowanie. Ta czesc funkcji odpowiada za prawidlowe posortowanie ciagu liczb ktore znajduja sie na kopcu. Czyli mierzemy najwiekszy element, odkladmy go w nowym wektorze, nastepnie zamieniamy ten element z ostatnim na kopcu i go usuwamy, po czym wywolujemy funkcje utowrzenia drzewa binarnego jeszcze raz. Elementy te powtrzamy tak dlugo, az w kopcu nie bedzie zadnej liczby do posortowania.
- void **quick** (**dane** ∗wektorQ, int a, int b)

  Funkcja sortujaca - quicksort Zadaniem tej funkcji jest posortowanie tablicy. Funkcja ta polega na metodzie dziel i zwyciezaj. Oznacza to ze ciag liczb dzielony jest na mniejsze wzgledem ustalonego elementu, nastepnie sortowane sa elementy po lewej stronie i prawej mediany poprzez rekursywne wyowalanie funkcji. Rekursja wystepuje do momentu podzielenia wektora na jednoelementowe czesci i laczenie ich w jedna calosc. Funkcji tej sa dwie wersje wyszukiwania mediany, poprzez pierwszy element wektora lub poprzez losowa wartosc [in] a - element wzgledem ktorego ma byc podzielony wektor.

4.6.1 Detailed Description

Klasa modeluje uruchomienia gownych wlasciowosci programu. Atrybutem klasy sa stowrzone dwa elemnty klasy dane, na ktorych wykonywane sa dzialania.

Definition at line 28 of file uruchom.hh.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 zegar::zegar ( ) [inline]

Definition at line 45 of file uruchom.hh.

4.6.3 Member Function Documentation

4.6.3.1 void zegar::algorytm ( )

Funkcja wykonujca zadany algorytm na wektorze. Funkcja wykonuje zadany algorytm na wektorze wejsciowym. W naszym przypadku wektor pomnoony jest przez sta liczb

Returns

(brak)

Definition at line 15 of file uruchom.cpp.

4.6.3.2 dane zegar::heap (dane *tab*)

Funkcja tworzaca kopiec Funkcja ta uzyta jest do sortowania poprzez kopcowanie. Tworzy drzewo binarne czyli kopiec Najwiekszy element jest na szczycie kopca i jest to tzw ojciec, ktory ma dwoch synow ktore sa mniejsze od niego, jezeli nie zamienia miejscami ojca z synem.Tak stowrzony kopiec uporzadkowany jest prawie malejaco.

Parameters

in	*tab*	- wektor z danymi, na ktorym bediemy operowac czyli robic kopiec.

Returns

tab - funkcja zwraca utworzony kopiec

Definition at line 122 of file uruchom.cpp.

4.6.3.3 dane zegar::heap_sort (dane *tab*)

Glowna funkcja sortujaca poprzez kopcowanie Funkcja sortujaca liczby poprzez kopcowanie. Ta czesc funkcji odpowiada za prawidlowe posortowanie ciagu liczb ktore znajduja sie na kopcu. Czyli mierzemy najwiekszy element, odkladmy go w nowym wektorze, nastepnie zamieniamy ten element z ostatnim na kopcu i go usuwamy, po czym wywolujemy funkcje utowrzenia drzewa binarnego jeszcze raz. Elementy te powtrzamy tak dlugo, az w kopcu nie bedzie zadnej liczby do posortowania.

Parameters

in	*tab*	- zmienna bedaca kopcem z ciagiem liczb

Returns

wynik - funkcja zwraca wekotr posortowany.

Definition at line 145 of file uruchom.cpp.

4.6.3.4 bool zegar::porownaj ( ) [inline]

Funkcja porjca dwa wektory. Funkcja porje dwa wektory, sprawdza czy wszystkie elemnty s ze sob r.

Returns

Funkcja zwraca true jezeli wektory sa jednakowe w przeciwnym wypadku zostaje wzrocony false.

Definition at line 83 of file uruchom.hh.

4.6.3.5 void zegar::quick (dane * *wektorQ,* int *a,* int *b*)

Funkcja sortujaca - quicksort Zadaniem tej funkcji jest posortowanie tablicy. Funkcja ta polega na metodzie dziel i zwyciezaj. Oznacza to ze ciag liczb dzielony jest na mniejsze wzgledem ustalonego elementu, nastepnie sortowane sa elementy po lewej stronie i prawej mediany poprzez rekursywne wyowalanie funkcji. Rekursja wystepuje do momentu podzielenia wektora na jednoelementowe czesci i laczenie ich w jedna calosc. Funkcji tej sa dwie wersje wyszukiwania mediany, poprzez pierwszy element wektora lub poprzez losowa wartosc [in] a - element wzgledem ktorego ma byc podzielony wektor.

Parameters

in	*b*	- liczba elemntow do posortowania
in	*wektor*	- wektor z ciagiem liczb do posortowania

Definition at line 165 of file uruchom.cpp.

4.6.3.6 dane zegar::scal (dane *lewo,* dane *prawo*)

Funkcja wykorzystana do sortowania przez scalanie Funkcja ta odpowiedzialana jest za scalanie dwoch wektorow w jeden. Porownuje dwa elementy i wiekszy elemnt dodaje do wektora.

Parameters

in	*lewo*	- wektor z ciagiem liczb do scalania(polaczenia w jeden)
in	*prawo*	- wektor z ciagiem liczb do scalenia(polaczenia w jeden)

Returns

rezultat - funkcja zwraca wektor uporzadkowany

Definition at line 80 of file uruchom.cpp.

4.6.3.7 dane zegar::scal_sort (dane *tab*)

Sortowanie przez scalanie glowne wywolanie funkcji Funkcja sortujaca liczby poprzez scalnie. Ta czesc funkcji dzieli ciag na dwa rowne podciagi, nastepnie kazda z czesci sortujemy, wywolujac rekurencyjnie ta funkcje. Gdy podciagi beda uporzadkowane scalilmy te male podciagi i powstanie nam ciag liczb posortowanych.

Parameters

in	*tab*	- zmienna bedaca wektorem, na ktorym umieszczone sa liczby do posortowania

Returns

rezultat - funkcja zwraca wekotr posortowany.

Definition at line 56 of file uruchom.cpp.

4.6.3.8 void zegar::wczytaj (string *nazwa*)

Funkcja wczytujca dane do stosu. Funkcja otwiera plik zdefiniowany w glownej funkcji przez uytkownika, sprawdza czy plik zosta otwarty, jeeli zosta otwarty wczytana jest liczba elementow pliku, nastepnie wczytywane sa wszystkie liczby do tablicy.

Parameters

in	*nazwa*	- zmienna, ktostaje wprowadzona przez uytkownika do programu

Returns

(brak)

Definition at line 39 of file uruchom.cpp.

4.6.3.9 void zegar::wczytaj_dane_pod (string *nazwa_pliku_pod*) [inline]

Wczytanie danych podstawowych Funkcja wczytuje dane, na ktorych wykonywany jest algorytm. Dane te sa glowna funkcja programu.

Parameters

in	*nazwa_pliku_pod*	- jest to zmienna ktora jest ciagiem znakow (nazwa pliku), ktory ma zostac otwarty.

Returns

(brak)

Definition at line 57 of file uruchom.hh.

4.6.3.10 void zegar::wczytaj_dane_spr (string *nazwa_pliku_spr*) [inline]

Wczytanie danych sprawdzajacych Funkcja wczytuje dane, na ktore zostaja porwonane z danymi na ktorych zostal wykonany algorytm.

Parameters

in	*nazwa_pliku_spr*	- jest to zmienna ktora jest ciagiem znakow (nazwa pliku), ktory ma zostac otwarty, w celu porownania.

Returns

(brak)

Definition at line 68 of file uruchom.hh.

4.6.3.11 LARGE_INTEGER zegar::wlaczStoper ( )

Funkcja zapamietujca czas poczatkowy. Funkcja naleca do biblioteki "windows.h", stoper zostaje wczony. Funkcja naleca do funkcji bool QueryPerformanceCounter(_out LARGE_INTEGER ∗lpPerformanceCount), funkcja ta zwraca wartosc niezerowa jeeli waczenie zako sikcesem, natomiast w przeciwnym wypadku zostanie wyrzucony bd i zwrartosc 0. Dla komputerltiprocesorowych nie ma znaczenia, ktest uywany, mog jedynie rnimalnie czasy.

Definition at line 21 of file uruchom.cpp.

4.6.3.12 LARGE_INTEGER zegar::wylaczStoper ( )

Funkcja zapamietujca czas koy. Funkcja naleca do biblioteki "windows.h", stoper zostaje wyczony, aby zosta zmierzony czas wykonania algorytmu w programie, poprzez odje czasu pocztkowego od czasu koego. Funkcja naleca do funkcji bool QueryPerformanceCounter(_out LARGE_INTEGER ∗lpPerformanceCount), funkcja ta zwraca wartosc niezerowa jeeli wyaczenie zako sikcesem, natomiast w przeciwnym wypadku zostanie wyrzucony bd i zwrartosc 0. Dla komputerltiprocesorowych nie ma znaczenia, ktest uywany, mog jedynie rnimalnie czasy.

Definition at line 30 of file uruchom.cpp.

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

- C:/Users/Ania/workspace/zadanie/inc/**uruchom.hh**
- C:/Users/Ania/workspace/zadanie/src/**uruchom.cpp**

5 File Documentation

5.1 C:/Users/Ania/workspace/zadanie/doc/pages/strona.dox File Reference

5.2 C:/Users/Ania/workspace/zadanie/inc/dane.hh File Reference

Definicja klasy dane.

```
#include <iostream>
#include <string>
#include <vector>
```

Include dependency graph for dane.hh: This graph shows which files directly or indirectly include this file:

Classes

- class **dane**

  Modeluje pojecie danych, uytych w programie, kt moim przypadku sa wektorem, ktory zostaje wczytany z pliku. Pierwsza zmienna wczytana z pliku jest liczba elemntow wystepujacych w tym pliku. Przyjelam, ze zmienne wczytane z pliku sa liczbami calkowitymi (int).

5.2.1 Detailed Description

Definicja klasy dane. Plik zawiera definicje klasy dane, ktora jest klasa podstawowa programu

Definition in file **dane.hh**.

5.3 C:/Users/Ania/workspace/zadanie/inc/kolejka.hh File Reference

Definicja klasy kolejkatab.

```
#include "uruchom.hh"
```

Include dependency graph for kolejka.hh: This graph shows which files directly or indirectly include this file:

Classes

- class **kolejkatab**

  Klasa modeluje pojecie kolejki, bazujacej na tablicy. Wykonywane sa operacje dodawnia do kolejki, zdjecia z kolejki, sprawdzenia czy jest pusta oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny. Dodatkowo uyte niektore funkcje pomocnicze jak np. powiekszenie tablicy.

5.3.1 Detailed Description

Definicja klasy kolejkatab. Plik zawiera definicje klasy kolejkatab, w ktorej wykorzystana jest tablica do zapisania elementow w kolejce.

Definition in file **kolejka.hh**.

5.4 C:/Users/Ania/workspace/zadanie/inc/kolejka_lista.hh File Reference

Definicja klasy kolejkalista.

```
#include <list>
#include <iostream>
#include <cstdlib>
```

Include dependency graph for kolejka_lista.hh: This graph shows which files directly or indirectly include this file:

Classes

- class **kolejkalista**

  Klasa modeluje pojecie kolejki, bazujacej na liscie. Wykonywane sa operacje dodawnia fo kolejki, zdjecia z kolejki, sprawdzenia czy jest pusta oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny.

5.4.1 Detailed Description

Definicja klasy kolejkalista. Plik zawiera definicje klasy kolejkalista, w ktorej wykorzystana jest lista do zapisania elementow w kolejce.

Definition in file **kolejka_lista.hh**.

5.5 C:/Users/Ania/workspace/zadanie/inc/stos.hh File Reference

Definicja klasy stos.

```
#include <iostream>
#include <cstdlib>
```

Include dependency graph for stos.hh: This graph shows which files directly or indirectly include this file:

Classes

- class **stos**

  Klasa modeluje pojecie stosu,bazujacego na tablicy. Na stosie wyonywane sa operacje dodatnia na stos, zdjecia ze stosu, sprawdzenia czy jest pusty oraz sprawdzenia rozmiaru. Tablice zaalokowalam w sposob dynamiczny. Dodatkowo uyte niektore funkcje pomocnicze jak np. powiekszenie stosu.

5.5.1 Detailed Description

Definicja klasy stos. Plik zawiera definicje klasy stos, ktora do zapisu i zapamietania liczb uzywa tablice.

Definition in file **stos.hh**.

5.6 C:/Users/Ania/workspace/zadanie/inc/stos_lista.hh File Reference

Definicja klasy stoslista.

```
#include <list>
#include <iostream>
#include <cstdlib>
```

Include dependency graph for stos_lista.hh: This graph shows which files directly or indirectly include this file:

Classes

- class **stoslista**

Klasa modeluje pojecie stosu,bazujacego na liscie. Na stosie wyonywane sa operacje dodatnia na stos, zdjecia ze stosu, sprawdzenia czy jest pusty oraz sprawdzenia rozmiaru.

5.6.1 Detailed Description

Definicja klasy stoslista. Plik zawiera definicje klasy stoslista, ktora do zapisu i zapamietania liczb uzywa listy.

Definition in file **stos_lista.hh**.

5.7 C:/Users/Ania/workspace/zadanie/inc/uruchom.hh File Reference

Definicja klasy zegar.

```
#include "dane.hh"
#include <windows.h>
#include "stos.hh"
```

Include dependency graph for uruchom.hh: This graph shows which files directly or indirectly include this file:

Classes

- class **zegar**

  Klasa modeluje uruchomienia gownych wlasciowosci programu. Atrybutem klasy sa stowrzone dwa elemnty klasy dane, na ktorych wykonywane sa dzialania.

5.7.1 Detailed Description

Definicja klasy zegar. Plik zawiera definicje klasy zegar, ktora jest klasa glowna programu. Klasa ta jest pochodna i specjalizacja klasy dane.

Definition in file **uruchom.hh**.

5.8 C:/Users/Ania/workspace/zadanie/src/dane.cpp File Reference

```
#include "dane.hh"
#include <fstream>
```

Include dependency graph for dane.cpp:

5.9 C:/Users/Ania/workspace/zadanie/src/kolejka.cpp File Reference

```
#include "kolejka.hh"
```

Include dependency graph for kolejka.cpp:

5.10 C:/Users/Ania/workspace/zadanie/src/kolejka_lista.cpp File Reference

```
#include "kolejka_lista.hh"
```

Include dependency graph for kolejka_lista.cpp:

5.11 C:/Users/Ania/workspace/zadanie/src/main.cpp File Reference

Funkcja glowna ktorej glownym zalozeniem jest wczytanie plikow z rozna wielkoscia elementow znajdujacych siliku, obliczenie sredniej wartosci czasu, w jakim zostaje wykonany algorytm (w naszym przypadku pomnozenie przez 2),nastepnie program porownuje poprawnosc wykoannia mnozenia z plikiem sprawdzajacym. Uzytkownik musi w

programie zdefiniowac: liczbe powtorzen (zmienna j), ilosc plikow - do ilu wykonywane jest mnozenie (zmienna i), nazwy plikow (string czesc_1, i, czesc_2 - wszystko opcjonalnie).

```
#include "uruchom.hh"
#include <sstream>
#include <fstream>
#include <time.h>
```

Include dependency graph for main.cpp:

Functions

- int **main** ()

5.11.1 Detailed Description

Funkcja glowna ktorej glownym zalozeniem jest wczytanie plikow z rozna wielkoscia elementow znajdujacych siliku, obliczenie sredniej wartosci czasu, w jakim zostaje wykonany algorytm (w naszym przypadku pomnozenie przez 2),nastepnie program porownuje poprawnosc wykoannia mnozenia z plikiem sprawdzajacym. Uzytkownik musi w programie zdefiniowac: liczbe powtorzen (zmienna j), ilosc plikow - do ilu wykonywane jest mnozenie (zmienna i), nazwy plikow (string czesc_1, i, czesc_2 - wszystko opcjonalnie).

Returns

(brak)

Definition in file **main.cpp**.

5.11.2 Function Documentation

5.11.2.1 int main ( )

Definition at line 59 of file main.cpp.

5.12 C:/Users/Ania/workspace/zadanie/src/stos.cpp File Reference

```
#include "stos.hh"
```

Include dependency graph for stos.cpp:

5.13 C:/Users/Ania/workspace/zadanie/src/stos_lista.cpp File Reference

```
#include "stos_lista.hh"
```

Include dependency graph for stos_lista.cpp:

5.14 C:/Users/Ania/workspace/zadanie/src/uruchom.cpp File Reference

```
#include "uruchom.hh"
#include <fstream>
#include <cmath>
#include <ctime>
```

Include dependency graph for uruchom.cpp:

Index