

Tablica asocjacyjna

Generated by Doxygen 1.8.1.2

Mon Apr 7 2014 00:10:35

Spis treści

1	Class Index	1
1.1	Class List	1
2	Class Documentation	3
2.1	Działanie Class Reference	3
2.1.1	Detailed Description	4
2.1.2	Member Function Documentation	4
2.1.2.1	heap	4
2.1.2.2	merge	4
2.1.2.3	MergeSort	4
2.1.2.4	Quicksort	4
2.1.2.5	Quicksort_lepiej	5
2.1.2.6	sprawdz	5
2.1.2.7	uporzadkuj_kopiec	5
2.1.2.8	wczytajDaneWejscowe	5
2.1.2.9	wlaczStoper	5
2.1.2.10	wykonajAlgorytm	6
2.1.2.11	wylaczStoper	6
2.2	kolejka_list Class Reference	6
2.2.1	Detailed Description	6
2.2.2	Member Function Documentation	6
2.2.2.1	dequeue	6
2.2.2.2	enqueue	7
2.2.2.3	isempty	7
2.2.2.4	size	7
2.3	kolejka_tab Class Reference	7
2.3.1	Detailed Description	8
2.3.2	Member Function Documentation	8
2.3.2.1	dequeue	8
2.3.2.2	enqueue	8
2.3.2.3	isempty	8

2.3.2.4	size	8
2.4	Kontener Class Reference	8
2.4.1	Detailed Description	9
2.4.2	Member Function Documentation	9
2.4.2.1	dodaj_element	9
2.4.2.2	dodaj_elementy	10
2.4.2.3	operator+	10
2.4.2.4	operator=	10
2.4.2.5	operator==	10
2.4.2.6	operator[]	10
2.4.2.7	wczytajDane	11
2.4.2.8	wez_dane	11
2.4.2.9	wez_rozmiar	11
2.4.2.10	zamien_elementy	11
2.5	Para< Wartosc > Class Template Reference	11
2.5.1	Detailed Description	12
2.5.2	Member Function Documentation	12
2.5.2.1	klucz	12
2.5.2.2	klucz	12
2.5.2.3	wartosc	12
2.5.2.4	wartosc	13
2.6	stos_list Class Reference	13
2.6.1	Detailed Description	13
2.6.2	Member Function Documentation	13
2.6.2.1	isempty	13
2.6.2.2	pop	13
2.6.2.3	push	14
2.6.2.4	size	14
2.7	stos_tab Class Reference	14
2.7.1	Detailed Description	14
2.7.2	Member Function Documentation	15
2.7.2.1	isempty	15
2.7.2.2	pop	15
2.7.2.3	push	15
2.7.2.4	size	15
2.8	Tablica_asocjacyjna< Wartosc > Class Template Reference	15
2.8.1	Detailed Description	16
2.8.2	Constructor & Destructor Documentation	16
2.8.2.1	Tablica_asocjacyjna	16
2.8.3	Member Function Documentation	16

2.8.3.1	czy_znalazlo	17
2.8.3.2	dodaj	17
2.8.3.3	usun	17
2.8.3.4	zmien	17
2.8.3.5	znajdz	18
2.9	Tester Class Reference	18
2.9.1	Detailed Description	18
2.9.2	Member Function Documentation	18
2.9.2.1	zamienNazwy	18

Rozdział 1

Class Index

1.1 Class List

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

Dzialanie	Klasa modelujaca gowna czesc programu	3
kolejka_list	Klasa modelujaca strukture koleki Kolejka jest zbudowana w oparciu o liste	6
kolejka_tab	Klasa modelujaca strukture koleki Kolejka jest zbudowana w oparciu o liste	7
Kontener	Klasa Dane	8
Para< Wartosc >	Klasa modeluje pare klucz-wartosc, ktora jest podstawowym elementem tablicy asocjacyjnej Zar typ Klucz jak i Wartosc musz miec konstruktory parametryczne, ktore inicjuja ich wartosci . . .	11
stos_list	Klasa modelujaca strukture stosu Stos jest zbudowany w oparciu o liste	13
stos_tab	Klasa modelujaca strukture stosu Stos jest zbudowany w oparciu o tablice	14
Tablica_asocjacyjna< Wartosc >	Klasa modeluje pojecie tablicy asocjacyjnej Tablica asocjacyjna jest to tablica, w ktorej do war- tosci mozemy sie odwolywac poprzez klucze. Odwolanie poprez indeks rowniez jest mozliwe .	15
Tester	Klasa Tester Klasa modeluje narzedzie do benchmarkowania algorytmu. Poprzez wielokrotne wynonywanie algorytmu mozemy wyciagnac sredni czas dzialania, co jest dokladniejszym po- miarem	18

Rozdział 2

Class Documentation

2.1 Działanie Class Reference

Klasa modelująca gówna część programu.

```
#include <dzialanie.hh>
```

Public Member Functions

- void [wczytajDaneWejscowe](#) (string nazwa)
metoda wczytuje dane do tablicy znajdujacej sie w zmiennej wejscie.
- LARGE_INTEGER [włączStoper](#) ()
Metoda uruchamia pomiar czasu.
- LARGE_INTEGER [wylaczStoper](#) ()
Metoda konczy pomiar czasu.
- void [wykonajAlgorytm](#) ()
Metoda wykonuje algorytm na danych wejscowych (tablicy)
- bool [sprawdz](#) ()
Metoda sprawdza poprawnosc algorytmu.
- int [uruchom](#) (string nazwa)
Metoda wykonuje jednorazowy test algorytmu Metoda: -włącza zegar -wykonuje algorytm -wylacza zegar -sprawdza poprawnosc algorytmu return czas wykonywania algorytmu w milisekundach.
- void [Quicksort](#) (Kontener *tab, int lewy, int prawy)
Metoda implementujca sortowanie szybkie.
- void [Quicksort_lepiej](#) (Kontener *tab, int lewy, int prawy)
Metoda implementujca usprawnione sortowanie szybkie Wybor piwotu jest losowy, co zmniejsza szanse na pojawienie sie przypadku pesymistycznego.
- [Kontener heap](#) (Kontener tab)
Metoda implementujca sortowanie przez kopcowanie.
- void [uporzadkuj_kopiec](#) (Kontener *kopiec)
Metoda przywraca sturkturpca jesli ta zostala zaburzona.
- [Kontener merge](#) (Kontener lewy, Kontener prawy)
Metoda implementujca scalanie dwoch tablic Scalanie czyli laczenie dwoch posortowanych tablic w jedna(rowniez posortowana)
- [Kontener MergeSort](#) (Kontener tab)
Metoda implementujca sortowanie przez scalanie.
- int [wez_rozmiar](#) ()

2.1.1 Detailed Description

Klasa modelujaca gówna czesc programu.

Klasa modeluje gówna czesc porgramu, ktorego zadaniem jest: -wczytanie danych -zmierzenie czasu dzialania algorytmu -sprawdzenie poprawnosci tego algorytmu, majac oczekiwany wynik

2.1.2 Member Function Documentation

2.1.2.1 Kontener Dzialanie::heap (Kontener *tab*)

Metoda implementujca sortowanie przez kopcowanie.

Parameters

<i>tab</i>	- tablica do posortowania
------------	---------------------------

Returns

metoda zwraca posortowana tablice

2.1.2.2 Kontener Dzialanie::merge (Kontener *lewy*, Kontener *prawy*)

Metoda implementujca scalanie dwoch tablic Scalanie czyli laczenie dwoch posortowanych tablic w jedna(rowniez posortowana)

Parameters

<i>lewy</i>	- element pierwszy
<i>prawy</i>	- element drugi

Returns

metoda zwraca scalony zbior

2.1.2.3 Kontener Dzialanie::MergeSort (Kontener *tab*)

Metoda implementujca sortowanie przez scalanie.

Parameters

<i>tab</i>	- tablica do posortowania
------------	---------------------------

Returns

metoda zwraca posortowana tablice

2.1.2.4 void Dzialanie::Quicksort (Kontener * *tab*, int *lewy*, int *prawy*)

Metoda implementujca sortowanie szybkie.

Parameters

<i>tab</i>	- wskaznik na tablice do posortowania
<i>lewy</i>	- indeks poczatku tablicy
<i>prawy</i>	- indeks konca tablicy

2.1.2.5 void Działanie::Quicksort_lepsiej (Kontener * *tab*, int *lewy*, int *prawy*)

Metoda implementująca usprawnione sortowanie szybkie. Wybór piwołu jest losowy, co zmniejsza szansę na pojawienie się przypadku pesymistycznego.

Parameters

<i>tab</i>	- wskaźnik na tablicę do posortowania
<i>lewy</i>	- indeks początku tablicy
<i>prawy</i>	- indeks końca tablicy

2.1.2.6 bool Działanie::sprawdz ()

Metoda sprawdza poprawność algorytmu.

Wczytywane są poprawne dane wyników, a następnie są one porównywane z tymi otrzymanymi przez wykonanie algorytmu.

Returns

0 - gdy algorytm jest poprawny, -1 - gdy nie.

2.1.2.7 void Działanie::uporządkuj_kopiec (Kontener * *kopiec*)

Metoda przywraca strukturę, jeśli ta została zaburzona.

Parameters

<i>kopiec</i>	- wskaźnik na kopiec do uporządkowania
---------------	--

2.1.2.8 void Działanie::wczytajDaneWejsciowe (string *nazwa*)

Metoda wczytuje dane do tablicy znajdującej się w zmiennej wejście.

Format danych w pliku jest następujący: pierwszy wiersz - ilość elementów, a następnie w kolumnie kolejne wartości tablicy.

Parameters

<i>nazwaPliku</i>	- nazwa pliku do otwarcia
-------------------	---------------------------

Returns

void

2.1.2.9 LARGE_INTEGER Działanie::włączStoper ()

Metoda uruchamia pomiar czasu.

Czas jest mierzony w milisekundach.

Returns

czas, w którym został włączony stoper

2.1.2.10 void Dzialanie::wykonajAlgorytm ()

Metoda wykonuje algorytm na danych wejsciowych (tablicy)

Algorytm do wykonania : pomnoz kazdy element razy 2.

Returns

void

2.1.2.11 LARGE_INTEGER Dzialanie::wylaczStoper ()

Metoda konczy pomiar czasu.

Czas jest mierzony w milisekundach

Returns

czas, w którym stoper został wyłączony

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

- inc/dzialanie.hh
- src/dzialanie.cpp

2.2 kolejka_list Class Reference

Klasa modelująca strukturę kolejki Kolejka jest zbudowana w oparciu o listę.

```
#include <kolejka_list.hh>
```

Public Member Functions

- void **enqueue** (int element)
Metoda dodająca element na stos Metoda dodaje element na koniec listy.
- void **dequeue** (int *a)
Metoda usuwająca element ze stos Metoda usuwa element znajdujący się na końcu listy. Gdy stos jest pusty, wyświetlony zostaje błąd mówiący o braku danych do ściągnięcia.
- bool **isempty** ()
Metoda sprawdzająca czy stos jest pusty. Metoda sprawdza czy stos jest pusty.
- int **size** ()
Metoda zwracająca rozmiar stosu Metoda zwraca rozmiar listy.
- void **wyswietl** ()

2.2.1 Detailed Description

Klasa modelująca strukturę kolejki Kolejka jest zbudowana w oparciu o listę.

2.2.2 Member Function Documentation

2.2.2.1 void kolejka_list::dequeue (int * a)

Metoda usuwająca element ze stos Metoda usuwa element znajdujący się na końcu listy. Gdy stos jest pusty, wyświetlony zostaje błąd mówiący o braku danych do ściągnięcia.

Parameters

<i>a</i>	- wskaznik do zmiennej, do ktorej sciagamy wartosc.
----------	---

2.2.2.2 void kolejka_list::enqueue (int *element*)

Metoda dodajaca element na stos Metoda dodaje element na koniec listy.

Parameters

<i>element</i>	- element, ktory zostanie dodany do stosu
----------------	---

2.2.2.3 bool kolejka_list::isempty ()

Metoda sprawdzajaca czy stos jest pusty. Metoda sprawdza czy stos jest pusty.

Returns

lista.empty();

2.2.2.4 int kolejka_list::size ()

Metoda zwracajaca rozmiar stosu Metoda zwraca rozmiar listy.

Returns

rozmiar stosu.

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

- inc/kolejka_list.hh
- src/kolejka_list.cpp

2.3 kolejka_tab Class Reference

Klasa modelujaca strukture koleki Kolejka jest zbudowana w oparciu o liste.

```
#include <kolejka_tab.hh>
```

Public Member Functions

- **kolejka_tab** (int *zwiększ*)
- void **enqueue** (int *element*)
Metoda dodajaca element na stos Metoda dodaje element na koniec listy.
- void **dequeue** (int **a*)
Metoda usuwajaca element ze stos Metoda usuwa element znajdujacy sie na koncu listy. Gdy stos jest pusty, wyswietlony zostaje blad mowiacy ze brak danych do sciagniecia.
- bool **isempty** ()
Metoda sprawdzajaca czy stos jest pusty. Metoda sprawdza czy stos jest pusty.
- int **size** ()
Metoda zwracajaca rozmiar stosu Metoda zwraca rozmiar listy.
- void **wyswietl** ()

2.3.1 Detailed Description

Klasa modelujaca strukture koleki Kolejka jest zbudowana w oparciu o liste.

2.3.2 Member Function Documentation

2.3.2.1 void kolejka_tab::dequeue (int * a)

Metoda usuwajaca element ze stos Metoda usuwa element znajdujacy sie na koncu listy. Gdy stos jest pusty, wyswietlony zostaje blad mowiacy ze brak danych do sciagniecia.

Parameters

<i>a</i>	- wskaznik do zmiennej, do ktorej sciagamy wartosc.
----------	---

2.3.2.2 void kolejka_tab::enqueue (int *element*)

Metoda dodajaca element na stos Metoda dodaje element na koniec listy.

Parameters

<i>element</i>	- element, ktory zostanie dodany do stosu
----------------	---

2.3.2.3 bool kolejka_tab::isempty ()

Metoda sprawdzajaca czy stos jest pusty. Metoda sprawdza czy stos jest pusty.

Returns

lista.empty();

2.3.2.4 int kolejka_tab::size ()

Metoda zwracajaca rozmiar stosu Metoda zwraca rozmiar listy.

Returns

rozmiar stosu.

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

- inc/kolejka_tab.hh
- src/kolejka_tab.cpp

2.4 Kontener Class Reference

Klasa Dane.

```
#include <kontener.hh>
```

Public Member Functions

- unsigned int [wez_rozmiar](#) ()
Metoda zwracająca rozmiar tablicy.
- vector< int > & [wez_dane](#) ()
Metoda zwracająca referencje do tablicy danych Metoda pozwala na dostęp do tablicy i jej modyfikacje.
- void [wczytajDane](#) (string nazwaPliku)
metoda wczytuje dane do tablicy z pliku
- void [zamien_elementy](#) (unsigned int i, unsigned int j)
Metoda zamienia ze sobą dwa elementy tablicy.
- void [odwroc_kolejnosc](#) ()
Metoda odwraca zawartość tablicy.
- void [dodaj_element](#) (int e)
Metoda dodaje element na koniec tablicy.
- void [usun_z_konca](#) ()
- void [usun_z_poczatku](#) ()
- void [dodaj_elementy](#) (Kontener tab)
Metoda dodaje na koniec tablicy zawartość innej tablicy.
- int & [operator\[\]](#) (int index)
Operator indeksujący tablice.
- const int & [operator\[\]](#) (int el) const
- Kontener & [operator+](#) (Kontener tab)
Operator dodawania tablic Operator pozwala na dodanie 2 tablic.
- Kontener & [operator=](#) (Kontener tab)
Operator przypisania Operator pozwala na przypisanie do tablicy zawartości innej tablicy.
- bool [operator==](#) (Kontener tab)
Operator porównania 2 tablic Operator pozwala na porównanie 2 tablic. Sprawdza on czy są takie same pod względem zawartości.

Friends

- ostream & [operator<<](#) (ostream &out, Kontener Tab)
Operator wypisywania Metoda pozwala na wypisanie zawartości tablicy na standardowe wyjście.

2.4.1 Detailed Description

Klasa Dane.

Klasa posiada 2 pola: -tablice (vector), -rozmiar tablicy.

2.4.2 Member Function Documentation

2.4.2.1 void Kontener::dodaj_element (int e)

Metoda dodaje element na koniec tablicy.

Parameters

<i>e</i>	- wartość elementu
----------	--------------------

2.4.2.2 void Kontener::dodaj_elementy (Kontener *tab*)

Metoda dodaje na koniec tablicy zawartosc innej tablicy.

Parameters

<i>tablica, ktora</i>	bedzie dodana na koniec
-----------------------	-------------------------

2.4.2.3 Kontener & Kontener::operator+ (Kontener *tab*)

Operator dodawania tablic Operator pozwala na dodanie 2 tablic.

`\param tablica do dodania`

Returns

dwie polaczone tablice

2.4.2.4 Kontener & Kontener::operator= (Kontener *tab*)

Operator przypisania Operator pozwala na przypisanie do tablicy zawartosci innej tablicy.

Parameters

<i>tablica, ktora</i>	przypisujemy
-----------------------	--------------

2.4.2.5 bool Kontener::operator== (Kontener *tab*)

Operator porownania 2 tablic Operator pozwala na porownanie 2 tablic. Sprawdza on czy sa takie same pod wzgle-
dem zawartosci.

Parameters

<i>tablica, z</i>	ktora bedziemy porownywac
-------------------	---------------------------

Returns

true - tablice sa identyczne

false - tablice sa rozne

2.4.2.6 int & Kontener::operator[] (int *index*)

Operator indeksujacy tablice.

Parameters

<i>index</i>	- indeks, ktorego referencja zostanie zwrzcona
--------------	--

Returns

referencja do zadanego indeksu

2.4.2.7 void Kontener::wczytajDane (string nazwaPliku)

metoda wczytuje dane do tablicy z pliku

Format danych w pliku jest następujący: pierwszy wiersz - ilość elementów, a następnie w kolumnie kolejne wartości tablicy.

Parameters

<i>nazwaPliku</i>	- nazwa pliku do otwarcia
-------------------	---------------------------

Returns

void

2.4.2.8 vector<int>& Kontener::wez_dane () [inline]

Metoda zwracająca referencje do tablicy danych Metoda pozwala na dostęp do tablicy i jej modyfikacje.

Returns

referencja do tablicy

2.4.2.9 unsigned int Kontener::wez_rozmiar () [inline]

Metoda zwracająca rozmiar tablicy.

Returns

rozmiar tablicy

2.4.2.10 void Kontener::zamien_elementy (unsigned int i, unsigned int j)

Metoda zamienia ze sobą dwa elementy tablicy.

Parameters

<i>i,j</i>	- indeksy, które zostaną zamienione
------------	-------------------------------------

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

- inc/kontener.hh
- src/kontener.cpp

2.5 Para< Wartosc > Class Template Reference

Klasa modeluje parę klucz-wartość, która jest podstawowym elementem tablicy asocjacyjnej. Zarówno typ Klucz jak i Wartość muszą mieć konstruktory parametryczne, które inicjują ich wartości.

```
#include <slownik.hh>
```

Public Member Functions

- [Para](#) ()

Konstruktor inicjujący wartosci key i val.

- [Para](#) (const std::string &key)

Konstruktor inicjujący wartosci key i val.

- [Para](#) (const std::string &key, const Wartosc &val)

Konstruktor inicjujący wartosci key i val.

- std::string & [klucz](#) ()

Metoda zwraca referencje do pola key, dzięki niej możemy modyfikować jego wartosc.

- Wartosc & [wartosc](#) ()

Metoda zwraca referencje do pola val, dzięki niej możemy modyfikować jego wartosc.

- const std::string & [klucz](#) () const

Metoda zwraca wartosc pola key, służy ona do odczytu tej wartosci.

- const Wartosc & [wartosc](#) () const

Metoda zwraca wartosc pola val, służy ona do odczytu tej wartosci.

2.5.1 Detailed Description

```
template<typename Wartosc>class Para< Wartosc >
```

Klasa modeluje parę klucz-wartosc, która jest podstawowym elementem tablicy asocjacyjnej. Zarówno typ Klucz jak i Wartosc muszą mieć konstruktory parametryczne, które inicjują ich wartości.

2.5.2 Member Function Documentation

2.5.2.1 `template<typename Wartosc> std::string& Para< Wartosc >::klucz ()` `[inline]`

Metoda zwraca referencję do pola key, dzięki niej możemy modyfikować jego wartosc.

Returns

referencja do pola key

2.5.2.2 `template<typename Wartosc> const std::string& Para< Wartosc >::klucz () const` `[inline]`

Metoda zwraca wartosc pola key, służy ona do odczytu tej wartosci.

Returns

wartosc pola key

2.5.2.3 `template<typename Wartosc> Wartosc& Para< Wartosc >::wartosc ()` `[inline]`

Metoda zwraca referencję do pola val, dzięki niej możemy modyfikować jego wartosc.

Returns

referencja do pola val

2.5.2.4 `template<typename Wartosc> const Wartosc& Para< Wartosc >::wartosc () const [inline]`

Metoda zwraca wartosc pola val, sluzy ona do odczytu tej wartosci.

Returns

wartosc pola val

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

- inc/slownik.hh

2.6 stos_list Class Reference

Klasa modelujaca strukture stosu Stos jest zbudowany w oparciu o liste.

```
#include <stos_list.hh>
```

Public Member Functions

- void **push** (int element)
Metoda dodajaca element na stos Metoda dodaje element na koniec listy.
- void **pop** (int *a)
Metoda usuwajaca element ze stos Metoda usuwa element znajdujacy sie na koncu listy. Gdy stos jest pusty, wyswietlony zostaje blad mowiacy ze brak danych do sciagniecia.
- bool **isempty** ()
Metoda sprawdzajaca czy stos jest pusty. Metoda sprawdza czy stos jest pusty.
- int **size** ()
Metoda zwracajaca rozmiar stosu Metoda zwraca rozmiar listy.
- void **wyswietl** ()

2.6.1 Detailed Description

Klasa modelujaca strukture stosu Stos jest zbudowany w oparciu o liste.

2.6.2 Member Function Documentation

2.6.2.1 bool stos_list::isempty ()

Metoda sprawdzajaca czy stos jest pusty. Metoda sprawdza czy stos jest pusty.

Returns

lista.empty();

2.6.2.2 void stos_list::pop (int * a)

Metoda usuwajaca element ze stos Metoda usuwa element znajdujacy sie na koncu listy. Gdy stos jest pusty, wyswietlony zostaje blad mowiacy ze brak danych do sciagniecia.

Parameters

a	- wskaznik do zmiennej, do ktorej sciagamy wartosc.
---	---

2.6.2.3 void stos_list::push (int *element*)

Metoda dodajaca element na stos Metoda dodaje element na koniec listy.

Parameters

<i>element</i>	- element, ktory zostanie dodany do stosu
----------------	---

2.6.2.4 int stos_list::size ()

Metoda zwracajaca rozmiar stosu Metoda zwraca rozmiar listy.

Returns

rozmiar stosu.

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

- inc/stos_list.hh
- src/stos_list.cpp

2.7 stos_tab Class Reference

Klasa modelujaca strukture stosu Stos jest zbudowany w oparciu o tablice.

```
#include <stos_tab.hh>
```

Public Member Functions

- **stos_tab** (int zwieksz)
- void **push** (int element)
Metoda dodajaca element na stos Metoda dodaje element na koniec tablicy. W przypadku gdy tablica jest pelna zostaje ona powiekszona o 1 element.
- void **pop** (int *a)
Metoda usuwajaca element ze stos Metoda usuwa element znajdujacy sie na koncu tablicy. Z kazdorazowym pobraniem danych rozmiar tablicy jest zmniejszany o 1.
- bool **isempty** ()
Metoda sprawdzajaca czy stos jest pusty. Metoda sprawdza czy stos jest pusty. Sprawdzenie polega na odczytaniu zmiennej rozmiar, ktora pamietala ile jest elementow na stosie.
- int **size** ()
Metoda zwracajaca rozmiar stosu Metoda zwraca rozmiar stosu. Rozmiar stosu przechowywany jest w zmiennej rozmiar.
- void **wyswietl** ()

2.7.1 Detailed Description

Klasa modelujaca strukture stosu Stos jest zbudowany w oparciu o tablice.

2.7.2 Member Function Documentation

2.7.2.1 bool stos_tab::isempty ()

Metoda sprawdzająca czy stos jest pusty. Metoda sprawdza czy stos jest pusty. Sprawdzenie polega na odczytaniu zmiennej rozmiar,

która pamięta ile jest elementów na stosie.

Returns

true - rozmiar == 0
false - rozmiar > 0.

2.7.2.2 void stos_tab::pop (int * a)

Metoda usuwająca element ze stosu Metoda usuwa element znajdujący się na końcu tablicy. Z każdorazowym pobraniem danych rozmiar tablicy jest zmniejszany o 1.

Parameters

<i>a</i>	- wskaźnik do zmiennej, do której ściągamy wartość.
----------	---

2.7.2.3 void stos_tab::push (int element)

Metoda dodająca element na stos Metoda dodaje element na koniec tablicy. W przypadku gdy tablica jest pełna zostaje ona powiększona o 1 element.

Parameters

<i>element</i>	- element, który zostanie dodany do stosu
----------------	---

2.7.2.4 int stos_tab::size ()

Metoda zwracająca rozmiar stosu Metoda zwraca rozmiar stosu. Rozmiar stosu przechowywany jest w zmiennej rozmiar.

Returns

rozmiar stosu.

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

- inc/stos_tab.hh
- src/stos_tab.cpp

2.8 Tablica_asocjacyjna< Wartosc > Class Template Reference

Klasa modeluje pojęcie tablicy asocjacyjnej Tablica asocjacyjna jest to tablica, w której do wartości możemy się odwoływać poprzez klucze. Odwołanie poprzez indeks również jest możliwe.

```
#include <slownik.hh>
```

Public Member Functions

- [Tablica_asocjacyjna](#) ()
Metoda dodaje pare klucz-wartosc do tablicy.
- void [dodaj](#) (std::string klucz, Wartosc wartosc)
Metoda dodaje pare klucz-wartosc do tablicy W przypadku gdy dany klucz juz istnieje to wartosc, ktora jest z nim skojarzona zostaje zastapiona ta podana w argumencie "wartosc" Po kazdorazowym dodaniu nowego klucza, tablica jest sortowana alfabetycznie wzgledem nazw kluczy. Takie rozwiazanie daje mozliwosc przeszukiwania binarnego co powoduje ze dostep do elementu jest w czasie logn.
- void [zmien](#) (string klucz, Wartosc wartosc)
Metoda pobiera wartosc przypisana pod zadany klucz.
- Wartosc [pobierz](#) (std::string klucz) const
- void [usun](#) (string klucz)
Metoda usuwa pozycje zawierajaca podany klucz. Gdy klucz istnieje, to jest usuwany, a flaga czy_istnieje ustawiana na wartosc true, przeciwnie flaga ma wartosc false.
- string [znajdz](#) (Wartosc wartosc)
Metoda znajduje pozycje(nazwe klucza), pod ktora znajduje sie podana wartosc.
- const bool [czy_znalazlo](#) () const
Metoda zwraca stan flagi czy_istnieje. Flaga czy istnieje jest ustawiana za kazdym razem gdy wykonujemy funkcje, ktora przeszukuje tablice po kluczach. Gdy zadany klucz istnieje to flaga jest ustawiana. Natomiast gdy danego klucza nie ma w zbiorze to flaga jest zerowana.
- const bool [czy_pusta](#) ()
- const int [rozmiar](#) ()
- Wartosc [operator\[\]](#) (string klucz) const
- Wartosc & [operator\[\]](#) (string klucz)

2.8.1 Detailed Description

```
template<typename Wartosc>class Tablica_asocjacyjna< Wartosc >
```

Klasa modeluje pojecie tablicy asocjacyjnej Tablica asocjacyjna jest to tablica, w ktorej do wartosci mozemy sie odwoływac poprzez klucze. Odwołanie poprzez indeks rowniez jest mozliwe.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 `template<typename Wartosc > Tablica_asocjacyjna< Wartosc >::Tablica_asocjacyjna () [inline]`

Metoda dodaje pare klucz-wartosc do tablicy.

zmienna pomocnicza, ktora sluzy do okreslania czy wartosc o podanym kluczu

znajduje sie w zbiorze

Parameters

<i>klucz</i>	
<i>wartosc</i>	

2.8.3 Member Function Documentation

2.8.3.1 `template<typename Wartosc > const bool Tablica_asocjacyjna< Wartosc >::czy_znalazlo () const`
`[inline]`

Metoda zwraca stan flagi czy_istnieje. Flaga czy istnieje jest ustawiana za kazdym razem gdy wykonujemy funkcje, ktora przeszukuje tablice po kluczach. Gdy zadany klucz istnieje to flaga jest ustawiana. Natomiast gdy danego klucza nie ma w zbiorze to flaga jest zerowana.

Returns

true - zadany klucz istnieje w zbiorze
false - klucz nie istnieje.

2.8.3.2 `template<typename Wartosc > void Tablica_asocjacyjna< Wartosc >::dodaj (std::string klucz, Wartosc wartosc)`

Metoda dodaje pare klucz-wartosc do tablicy W przypadku gdy dany klucz juz istnieje to wartosc, ktora jest z nim skojarzona zostaje zastapiona ta

podana w argumencie "wartosc" Po kazdorazowym dodaniu nowego klucza, tablica jest sortowana alfabetycznie wzgledem nazw kluczy. Takie rozwiazanie daje mozliwosc przeszukiwania binarnego co powoduje ze dostep do elementu jest w czasie logn.

Parameters

in	<i>klucz-nazwa</i>	klucza
in	<i>wartosc-wartosc</i>	do zapisania

2.8.3.3 `template<typename Wartosc > void Tablica_asocjacyjna< Wartosc >::usun (string klucz)`

Metoda usuwa pozycje zawierajaca podany klucz. Gdy klucz istnieje, to jest usuwany, a flaga czy_istnieje ustawiana na wartosc true, przeciwnie flaga ma wartosc false.

Parameters

in	<i>klucz-klucz</i>	do usuniecia
----	--------------------	--------------

2.8.3.4 `template<typename Wartosc > void Tablica_asocjacyjna< Wartosc >::zmien (string klucz, Wartosc wartosc)`
`[inline]`

Metoda pobiera wartosc przypisana pod zadany klucz.

Jesli klucz nie istnieje to flaga czy_istnieje jest ustawiana na false, w przeciwnym wypadku ma ona wartosc true. Po wywołaniu metody "pobierz" mozna sprawdzic stan flagi metoda [czy_znalazlo\(\)](#) i jego podstawie okreslic czy operacja zostala wykonana poprawnie.

Parameters

in	<i>klucz-</i>	nazwa klucza, ktorego szukamy
----	---------------	-------------------------------

Returns

wartosc spod zadanego klucza

2.8.3.5 `template<typename Wartosc > string Tablica_asocjacyjna< Wartosc >::znajdz (Wartosc wartosc)`

Metoda znajduje pozycje(nazwe klucza), pod ktora znajduje sie podana wartosc.

Returns

gdy wartosc istnieje: nazwa klucza,
gdy nie istnieje: pusta zmienna string

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

- inc/slownik.hh

2.9 Tester Class Reference

Klasa [Tester](#) Klasa modeluje narzedzie do benchmarkowania algorytmu. Poprzez wielokrotne wynonywanie algorytmu mozemy wyciagnac sredni czas dzialania, co jest dokladniejszym pomiarem.

```
#include <tester.hh>
```

Public Member Functions

- [Tester](#) ()
konstruktor klasy Konstruktor inicjuje wartosci: -powtorzenia (ile razy wykonywac algorytm) -ilosc (ile mamy zestawow dancyh) -wejście (nazwa pliku z danymi wejsciowymi) -wynik (nazwa pliku z poprawnym wynikiem algorytmu)
- void [otworzPlik](#) (string nazwa)
Metoda otwierajaca referencje do pliku CSV.
- void [zamknijPlik](#) ()
Metoda otwierajaca referencje do pliku CSV.
- void [symulacja](#) ()
Metoda symulujaca badanie algorytmu Metoda wykonuje symulacje dzialania algorytmu. Wykonuje algorytm zadana liczbe razy dla zadanych zestawow danych. Wyniki zapisuje do pliku CSV o nazwie rezultat.csv Format zapisu: rozmiar_problemu,ilosc_powtorzen,sredni_czas.
- void [zamienNazwy](#) (int numer)
Metoda aktualizuje nazwe plikow wejsciowych Format plikow wejsciowych jest scisle okreslony. Dane wejsciowe: wejście%.txt, gdzie % to numer zestawu Dane wynikowe: wynik%.txt, gdzie % to numer zestawu Zestawy nalazy numerowac od 0.

2.9.1 Detailed Description

Klasa [Tester](#) Klasa modeluje narzedzie do benchmarkowania algorytmu. Poprzez wielokrotne wynonywanie algorytmu mozemy wyciagnac sredni czas dzialania, co jest dokladniejszym pomiarem.

2.9.2 Member Function Documentation

2.9.2.1 void [Tester::zamienNazwy](#) (int numer)

Metoda aktualizuje nazwe plikow wejsciowych Format plikow wejsciowych jest scisle okreslony. Dane wejsciowe: wejście%.txt, gdzie % to numer zestawu Dane wynikowe: wynik%.txt, gdzie % to numer zestawu Zestawy nalazy numerowac od 0.

Parameters

<i>numer</i>	- % zostaje zamieniony na wartosc 'numer'
--------------	---

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

- inc/tester.hh
- src/tester.cpp

Skorowidz

czy_znalazlo
 Tablica_asocjacyjna, 16

dequeue
 kolejka_list, 6
 kolejka_tab, 8

dodaj
 Tablica_asocjacyjna, 17

dodaj_element
 Kontener, 9

dodaj_elementy
 Kontener, 9

Dzialanie, 3
 heap, 4
 merge, 4
 MergeSort, 4
 Quicksort, 4
 Quicksort_lepiej, 5
 sprawdz, 5
 uporzadkuj_kopiec, 5
 wczytajDaneWejscowe, 5
 włączStoper, 5
 wykonajAlgorytm, 5
 wylączStoper, 6

enqueue
 kolejka_list, 7
 kolejka_tab, 8

heap
 Dzialanie, 4

isempty
 kolejka_list, 7
 kolejka_tab, 8
 stos_list, 13
 stos_tab, 15

klucz
 Para, 12

kolejka_list, 6
 dequeue, 6
 enqueue, 7
 isempty, 7
 size, 7

kolejka_tab, 7
 dequeue, 8
 enqueue, 8
 isempty, 8
 size, 8

Kontener, 8
 dodaj_element, 9
 dodaj_elementy, 9
 operator+, 10
 operator=, 10
 operator==, 10
 wczytajDane, 10
 wez_dane, 11
 wez_rozmiar, 11
 zamien_elementy, 11

merge
 Dzialanie, 4
MergeSort
 Dzialanie, 4

operator+
 Kontener, 10
operator=
 Kontener, 10
operator==
 Kontener, 10

Para
 klucz, 12
 wartosc, 12
Para< Wartosc >, 11

pop
 stos_list, 13
 stos_tab, 15

push
 stos_list, 14
 stos_tab, 15

Quicksort
 Dzialanie, 4
Quicksort_lepiej
 Dzialanie, 5

size
 kolejka_list, 7
 kolejka_tab, 8
 stos_list, 14
 stos_tab, 15

sprawdz
 Dzialanie, 5

stos_list, 13
 isempty, 13
 pop, 13
 push, 14

- size, [14](#)
- stos_tab, [14](#)
 - isempty, [15](#)
 - pop, [15](#)
 - push, [15](#)
 - size, [15](#)
- Tablica_asocjacyjna
 - czy_znalazlo, [16](#)
 - dodaj, [17](#)
 - Tablica_asocjacyjna, [16](#)
 - Tablica_asocjacyjna, [16](#)
 - usun, [17](#)
 - zmien, [17](#)
 - znajdz, [17](#)
- Tablica_asocjacyjna< Wartosc >, [15](#)
- Tester, [18](#)
 - zamienNazwy, [18](#)
- uporzadkuj_kopiec
 - Dzialanie, [5](#)
- usun
 - Tablica_asocjacyjna, [17](#)
- wartosc
 - Para, [12](#)
- wczytajDane
 - Kontener, [10](#)
- wczytajDaneWejscowe
 - Dzialanie, [5](#)
- wez_dane
 - Kontener, [11](#)
- wez_rozmiar
 - Kontener, [11](#)
- wlaczStoper
 - Dzialanie, [5](#)
- wykonajAlgorytm
 - Dzialanie, [5](#)
- wylaczStoper
 - Dzialanie, [6](#)
- zamien_elementy
 - Kontener, [11](#)
- zamienNazwy
 - Tester, [18](#)
- zmien
 - Tablica_asocjacyjna, [17](#)
- znajdz
 - Tablica_asocjacyjna, [17](#)