Tablica asocjacyjna

Wygenerowano przez Doxygen 1.8.4

Cz, 24 kwi 2014 00:23:40

Spis treści

1 Tablica asocjacyjne

1.1 Opis programu

Program umożliwiający operacje na tablicy asocjacyjnej - abstrakcyjnego typu danych, który przechowuje pary - unikatowy klucz oraz wartość. Tablica została zaimplementowana w trzy różne sposoby - korzystając z listy, drzewa binarnego oraz tablicy haszującej.

1.2 Sposób wywołania programu:

./asarray [aarray/hash/tree] [rozmiar problemu] [liczba powtórzeń]

1.3 Autor

Program wykonala: Agnieszka Wisniewska, nr albumu: 200 466

2 Indeks klas

2.1 Lista klas

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

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy Aarray< Value >

```
#include <aarray.hh>
```

Komponenty

- struct Pair

Metody publiczne

- void Add (std::string key, Value value)

  Funkcja dodająca parę klucz-wartość do tablicy.
- void Delete (std::string key)

  Funkcja usuwająca wybraną parę klucz-wartość z tablicy.
- Value GetValue (std::string key)

  Funkcja pobierająca wartość wybranego klucza.
- void ChangeValue (std::string key, Value newValue)

  Funkcja zmieniająca wartość pod wybranym kluczem.

- unsigned int Size ()

  Funkcja zwraca ilość par klucz-wartość
- bool IsEmpty ()

  Funkcja sprawdzająca, czy tablica jest pusta.

Metody prywatne

- unsigned int Search (std::string key)

Atrybuty prywatne

- std::vector< Pair > PairVec

4.1.1 Dokumentacja funkcji składowych

4.1.1.1 template<class Value> void Aarray< Value >::Add (std::string *key*, Value *value*)

Funkcja dodająca parę klucz-wartość do tablicy.

Parametry

value	wartość dla dodawanego klucza
key	unikatowy klucz, do którego przypisujemy wartość

4.1.1.2 template<class Value> void Aarray< Value >::ChangeValue (std::string *key*, Value *newValue*)

Funkcja zmieniająca wartość pod wybranym kluczem.

Parametry

newValue	nowa wartość dla klucza
key	unikatowy klucz, którego wartość zmieniamy

4.1.1.3 template<class Value > void Aarray< Value >::Delete (std::string *key*)

Funkcja usuwająca wybraną parę klucz-wartość z tablicy.

Parametry

key	klucz, który usuwamy wraz z jego wartością

4.1.1.4 template<class Value > Value Aarray< Value >::GetValue (std::string *key*)

Funkcja pobierająca wartość wybranego klucza.

Parametry

key	klucz, którego wartość chcemy znać

4.1.1.5 template<class Value > bool Aarray< Value >::IsEmpty ( )

Funkcja sprawdzająca, czy tablica jest pusta.

Zwraca

true jeśli tablica jest pusta
false jeśli na tablicy są jakieś pary klucz-wartość

4.1.1.6 template<class Value > unsigned int Aarray< Value >::Search (std::string *key*) [private]

4.1.1.7 template<class Value > unsigned int Aarray< Value >::Size ( )

Funkcja zwraca ilość par klucz-wartość

4.1.2 Dokumentacja atrybutów składowych

4.1.2.1 template<class Value> std::vector<Pair> Aarray< Value >::PairVec [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- aarray.hh

4.2 Dokumentacja klasy Benchmark

Definicja klasy Benchmark.

`#include <benchmark.hh>`

Metody publiczne

- double benchmark (int noIteration, sType type, int problemSize)

  Funkcja mierzaca czas wykonywania algorytmu sortowania.

Metody prywatne

- void Calculate (sType type)

Atrybuty prywatne

- Aarray< int > * aarrayptr
- Hash< int > * hashptr
- Tree< int > * treeptr
- std::string Tmp

4.2.1 Opis szczegółowy

Definicja klasy Benchmark.

Glownym zadaniem klasy jest mierzenie czasu wykonywanych algorytmow. Klasa obiektow umozliwiaja także wykonywanie zadanych obliczen.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 double Benchmark::benchmark (int *noIteration,* sType *type,* int *problemSize*)

Funkcja mierzaca czas wykonywania algorytmu sortowania.

4.2.2.2 void Benchmark::Calculate (sType *type*) [private]

4.2.3 Dokumentacja atrybutów składowych

4.2.3.1 Aarray<int>* Benchmark::aarrayptr [private]

4.2.3.2 Hash<int>* Benchmark::hashptr [private]

4.2.3.3 std::string Benchmark::Tmp [private]

4.2.3.4 Tree<int>* Benchmark::treeptr [private]

Dokumentacja dla tej klasy została wygenerowana z plików:

- benchmark.hh
- benchmark.cpp

4.3 Dokumentacja szablonu klasy Hash< Value >

Szablon klasy Hash.

`#include <hash.hh>`

Komponenty

- struct Pair

Metody publiczne

- Hash (const unsigned int problemSize)

  Konstruktor klasy Hash.
- void Add (const std::string &key, const Value &value)

  Funkcja dodająca parę klucz-wartość do tablicy.
- int Size () const

  Funkcja zwraca ilość par klucz-wartość
- bool Empty () const

  Funkcja sprawdzająca, czy tablica jest pusta.
- Value & operator[] (const std::string &key)

  Operator indeksujący tablicę
- const Value & operator[] (const std::string &key) const

Statyczne metody prywatne

- static unsigned long hash (const std::string &key)

Atrybuty prywatne

- unsigned int currentSize
- std::vector< std::list< Pair > > arr

4.3.1 Opis szczegółowy

template<class Value>class Hash< Value >

Szablon klasy Hash.

Definiuje funkcje pozwalające na wykonywanie operacji na tablicy asocjacyjnej stworzonej na tablicy haszującej. Tablica ta przechowuje dane indeksowane unikatowymi kluczami typu string (ze standardowej biblioteki)

Parametry

Value	typ wartości przechowywanych danych

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 template<class Value> Hash< Value >::Hash (const unsigned int *problemSize*) [inline]

Konstruktor klasy Hash.

Parametry

problemSize	docelowy rozmiar problemu

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 template<class Value> void Hash< Value >::Add (const std::string & *key*, const Value & *value*)

Funkcja dodająca parę klucz-wartość do tablicy.

Parametry

value	wartość dla dodawanego klucza
key	unikatowy klucz, do którego przypisujemy wartość

4.3.3.2 template<class Value> bool Hash< Value >::Empty ( ) const [inline]

Funkcja sprawdzająca, czy tablica jest pusta.

Zwraca

true jeśli tablica jest pusta
false jeśli na tablicy są jakieś pary klucz-wartość

4.3.3.3 template<class Value > unsigned long Hash< Value >::hash (const std::string & *key*) [static], [private]

4.3.3.4 template<class Value > Value & Hash< Value >::operator[] (const std::string & *key*)

Operator indeksujący tablicę

Parametry

key	klucz wybranego elementu

Zwraca

referencja na wybrany element

4.3.3.5 template<class Value > const Value & Hash< Value >::operator[] (const std::string & *key*) const

4.3.3.6 template<class Value> int Hash< Value >::Size ( ) const [inline]

Funkcja zwraca ilość par klucz-wartość

Zwraca

aktualny rozmiar tablicy

4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 template<class Value> std::vector<std::list<Pair> > Hash< Value >::arr [private]

4.3.4.2 template<class Value> unsigned int Hash< Value >::currentSize [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- hash.hh

4.4 Dokumentacja struktury Aarray< Value >::Pair

Atrybuty publiczne

- std::string key
- Value value

4.4.1 Dokumentacja atrybutów składowych

4.4.1.1 template<class Value> std::string Aarray< Value >::Pair::key

4.4.1.2 template<class Value> Value Aarray< Value >::Pair::value

Dokumentacja dla tej struktury została wygenerowana z pliku:

- aarray.hh

4.5 Dokumentacja struktury Hash< Value >::Pair

Atrybuty publiczne

- std::string key
- Value value

4.5.1 Dokumentacja atrybutów składowych

4.5.1.1 template<class Value> std::string Hash< Value >::Pair::key

4.5.1.2 template<class Value> Value Hash< Value >::Pair::value

Dokumentacja dla tej struktury została wygenerowana z pliku:

- hash.hh

4.6 Dokumentacja szablonu klasy Tree< Value >

Szablon klasy Hash.

`#include <tree.hh>`

Komponenty

- class TreeElement

Metody publiczne

- Tree ()

  Konstruktor klasy Tree.
- ~Tree ()

  Destruktor klasy Tree.
- void Add (const std::string &key, const Value &value)

  Funkcja dodająca parę klucz-wartość
- int Size () const

  Funkcja zwraca ilość par klucz-wartość
- bool isEmpty () const

  Funkcja sprawdzająca, czy tablica jest pusta.
- Value & operator[] (const std::string &key)

  Operator indeksujący tablicę
- const Value & operator[] (const std::string &key) const

Metody prywatne

- void * find (const std::string &key) const

Statyczne metody prywatne

- static void Deletee (TreeElement *pointer)

Atrybuty prywatne

- unsigned int arrSize
- TreeElement * root

4.6.1 Opis szczegółowy

template<class Value>class Tree< Value >

Szablon klasy Hash.

Definiuje funkcje pozwalające na wykonywanie operacji na tablicy asocjacyjnej stworzonej na drzewie binarnym.

Parametry

Value	typ wartości przechowywanych danych

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 template<class Value> Tree< Value >::Tree ( ) [inline]

Konstruktor klasy Tree.

4.6.2.2 template<class Value > Tree< Value >::~Tree ( )

Destruktor klasy Tree.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 template<class Value> void Tree< Value >::Add (const std::string & *key,* const Value & *value*)

Funkcja dodająca parę klucz-wartość

Parametry

value	wartość dla dodawanego klucza
key	unikatowy klucz, do którego przypisujemy wartość

4.6.3.2 template<class Value > void Tree< Value >::Deletee (TreeElement * *pointer*) [static], [private]

4.6.3.3 template<class Value > void * Tree< Value >::find (const std::string & *key*) const [private]

4.6.3.4 template<class Value> bool Tree< Value >::isEmpty ( ) const [inline]

Funkcja sprawdzająca, czy tablica jest pusta.

Zwraca

true jeśli tablica jest pusta
false jeśli na tablicy są jakieś pary klucz-wartość

4.6.3.5 template<class Value > Value & Tree< Value >::operator[] (const std::string & *key*)

Operator indeksujący tablicę

Parametry

key	klucz wybranego elementu

Zwraca

referencja na wybrany element

4.6.3.6 template<class Value > const Value & Tree< Value >::operator[] (const std::string & *key*) const

4.6.3.7 template<class Value> int Tree< Value >::Size ( ) const [inline]

Funkcja zwraca ilość par klucz-wartość

Zwraca

aktualny rozmiar tablicy

4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 template<class Value> unsigned int Tree< Value >::arrSize [private]

4.6.4.2 template<class Value> TreeElement* Tree< Value >::root [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- tree.hh

4.7 Dokumentacja klasy Tree< Value >::TreeElement

Metody publiczne

- TreeElement (const std::string &newKey, const Value &newValue)

Atrybuty publiczne

- std::string key
- Value value
- TreeElement * left
- TreeElement * right

4.7.1 Dokumentacja konstruktora i destruktora

4.7.1.1 template<class Value> Tree< Value >::TreeElement::TreeElement (const std::string & *newKey*, const Value & *newValue*) [inline]

4.7.2 Dokumentacja atrybutów składowych

4.7.2.1 template<class Value> std::string Tree< Value >::TreeElement::key

4.7.2.2 template<class Value> TreeElement* Tree< Value >::TreeElement::left

4.7.2.3 template<class Value> TreeElement* Tree< Value >::TreeElement::right

4.7.2.4 template<class Value> Value Tree< Value >::TreeElement::value

Dokumentacja dla tej klasy została wygenerowana z pliku:

- tree.hh

5 Dokumentacja plików

5.1 Dokumentacja pliku aarray.hh

Plik zawiera definicje szablonu klasy Aarray.

```
#include <vector>
#include <string>
#include <stdexcept>
```

Komponenty

- class Aarray< Value >
- struct Aarray< Value >::Pair

5.1.1 Opis szczegółowy

Plik zawiera definicje szablonu klasy Aarray.

5.2 Dokumentacja pliku benchmark.cpp

Plik zawiera implementacje klasy Benchmark.

```
#include <iostream>
#include <fstream>
#include <ctime>
#include "benchmark.hh"
#include <sstream>
#include <string>
#include <cstdlib>
```

5.2.1 Opis szczegółowy

Plik zawiera implementacje klasy Benchmark.

5.3 Dokumentacja pliku benchmark.hh

Plik zawiera definicje klasy Benchmark oraz typu sType.

```
#include "aarray.hh"
#include "hash.hh"
#include "tree.hh"
#include <string>
```

Komponenty

- class Benchmark

  Definicja klasy Benchmark.

Wyliczenia

- enum sType { aarray, hash, tree }

  Typ danych przechowujacy poszczególne struktury danych.

5.3.1 Opis szczegółowy

Plik zawiera definicje klasy Benchmark oraz typu sType.

5.3.2 Dokumentacja typów wyliczanych

5.3.2.1 enum sType

Typ danych przechowujacy poszczególne struktury danych.

Wartości wyliczeń

aarray

hash

tree

5.4 Dokumentacja pliku hash.hh

Plik zawiera definicje szablonu klasy Hash.

```
#include <stdexcept>
#include <string>
#include <vector>
#include <list>
```

Komponenty

- class Hash< Value >

  Szablon klasy Hash.
- struct Hash< Value >::Pair

5.4.1 Opis szczegółowy

Plik zawiera definicje szablonu klasy Hash.

5.5 Dokumentacja pliku main.cpp

Plik zawierający główną funkcję programu.

```
#include "benchmark.hh"
#include <iostream>
#include <cstdlib>
#include <ctime>
```

Funkcje

- int main (int argc, char **argv)

  Główna funkcja programu.

5.5.1 Opis szczegółowy

Plik zawierający główną funkcję programu.

5.5.2 Dokumentacja funkcji

5.5.2.1 int main (int *argc,* char ** *argv*)

Główna funkcja programu.

Pozwala na zmierzenie czasu dla poszczególnych implementacji tablicy asocjacyjnej na różnych strukturach danych.

5.6 Dokumentacja pliku mainpage.dox

5.7 Dokumentacja pliku tree.hh

Plik zawiera definicje szablonu klasy Tree.

```
#include <stdexcept>
#include <string>
```

Komponenty

- class Tree< Value >

  Szablon klasy Hash.
- class Tree< Value >::TreeElement

5.7.1 Opis szczegółowy

Plik zawiera definicje szablonu klasy Tree.

Skorowidz