PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Śr, 20 maj 2015 11:08:31

Spis treści

1 Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

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:

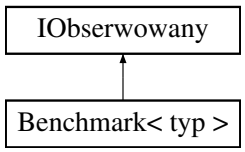
4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy Benchmark< typ >

Modeluje pojęcie Benchmarku.

`#include <Benchmark.hh>`

Diagram dziedziczenia dla Benchmark< typ >



Metody publiczne

- Benchmark (const unsigned int ileProb, unsigned int *const ileDanych, const unsigned int ilePowtorzen)

  Konstruktor 2 argumentowy.
- void Test (ITestable &I, const std::string &nazwaPliku)

  Testowanie algorytmu.
- void DodajObserwatora (IObserwator *nowyObserwator)

Dodaje Obserwatora.

- void UsunObserwatora (IObserwator ∗obserwator)

  Usuwa Obserwatora.

- void PowiadomObserwatorow ()

  Powiadamia Obserwatorów.

Atrybuty prywatne

- unsigned int IleProb

  Ilość prób.

- unsigned int ∗ IleDanych

  Tablica liczności serii.

- unsigned int IlePowtorzen

  Ilość powtórzeń

- std::list< IObserwator ∗ > ListaObserwatorow

  Lista Obserwatorow.

4.1.1 Opis szczegółowy

template<class typ>class Benchmark< typ >

Modeluje pojęcie Benchmarku czyli objektu mierzącego czas wykonywania algoytmu

Definicja w linii 26 pliku Benchmark.hh.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 template<class typ> Benchmark< typ >::Benchmark (const unsigned int *ileProb,* unsigned int ∗const *ileDanych,* const unsigned int *ilePowtorzen*) [inline]

Tworzy objekt klasy Benchmark i inicjuje nową statystykę dla objektu

Parametry

in	*ileProb*	- ilość prób, które zostaną wykonane
in	*ileDanych*	- wkaźnik na tablice z licznościami kolejnych serii
in	*ilePowtorzen*	- ilość powtórzeń każdej serii

Definicja w linii 71 pliku Benchmark.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 template<class typ> void Benchmark< typ >::DodajObserwatora (IObserwator ∗ *nowyObserwator*) [inline], [virtual]

Dodaje obserwatora do listy obserwatorów danego objektu

Parametry

in	*nowyObserwator*	- wskaźnik na objekt będący obserwatorem

Implementuje IObserwowany.

Definicja w linii 113 pliku Benchmark.hh.

4.1.3.2 template<class typ> void Benchmark< typ >::PowiadomObserwatorow ( ) [inline], [virtual]

Wywołuje u wszystkich aktywnych obserwatorów metodę Aktualizuj.

Implementuje IObserwowany.

Definicja w linii 133 pliku Benchmark.hh.

4.1.3.3 template<class typ> void Benchmark< typ >::Test (ITestable & *I*, const std::string & *nazwaPliku*) [inline]

Metoda testuje algorytm w okreslonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego pez użytkownika

Parametry

in	*I*	- objekt klasy implementującej ITestable na której zostanie przeprowadzony test
in	*nazwaPliku*	- nazwa pliku z danymi do wczytania

Definicja w linii 87 pliku Benchmark.hh.

4.1.3.4 template<class typ> void Benchmark< typ >::UsunObserwatora (IObserwator * *obserwator*) [inline], [virtual]

Usuwa danego obserwatora z listy obserwatorów

Parametry

in	*obserwator*	- wskaźnik na obserwatora który ma zostać usunięty

Implementuje IObserwowany.

Definicja w linii 124 pliku Benchmark.hh.

4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 template<class typ> unsigned int* Benchmark< typ >::IleDanych [private]

Tablica z licznościami elementów dla kojenych serii

Definicja w linii 42 pliku Benchmark.hh.

4.1.4.2 template<class typ> unsigned int Benchmark< typ >::IlePowtorzen [private]

Ilość powtórzeń każdej serii

Definicja w linii 50 pliku Benchmark.hh.

4.1.4.3 template<class typ> unsigned int Benchmark< typ >::IleProb [private]

Ilość powtórzeń każdej seriii

Definicja w linii 34 pliku Benchmark.hh.

4.1.4.4 template<class typ> std::list<IObserwator*> Benchmark< typ >::ListaObserwatorow [private]

Lista aktywnych obserwatorów danego objektu

Definicja w linii 57 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

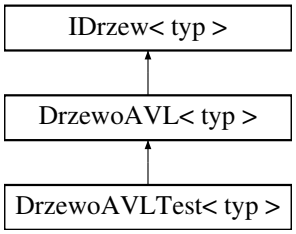
- Benchmark.hh

4.2 Dokumentacja szablonu klasy DrzewoAVL< typ >

Definicja drzewa AVL.

`#include <DrzewoAVL.hh>`

Diagram dziedziczenia dla DrzewoAVL< typ >



Metody publiczne

- DrzewoAVL ()
  Konstruktor bezarumentowy.
- ~DrzewoAVL ()
  Destruktor.
- void Insert (typ wartosc)
  Insert.
- void Remove (typ wartosc)
  Remove.
- WezelAVL< typ > * Search (typ wartosc)
  Search.
- void CzyscDrzewo ()
  CzyscDrzewo.

Metody prywatne

- WezelAVL< typ > * FindMin (WezelAVL< typ > *poszukiwacz)
  FindMin.
- WezelAVL< typ > * FindSuccessor (WezelAVL< typ > *poszukiwacz)
  FindSuccessor.
- void RR (WezelAVL< typ > *A)
  Rotacja RR.
- void LL (WezelAVL< typ > *A)
  Rotacja LL.
- void RL (WezelAVL< typ > *A)
  Rotacja RL.
- void LR (WezelAVL< typ > *A)
  Rotacja LR.
- void Czysc (Wezel< typ > *wezel)
  Czyści drzewo.

Atrybuty prywatne

- WezelAVL< typ > * Korzen
  Korzeń drzewa.
- int LiczbaWezlow
  Liczba Węzłów.

4.2.1 Opis szczegółowy

template<class typ>class DrzewoAVL< typ >

Pik zawiera definicję drzewa AVL.

The DrzewoAVL class

Klasa modeluje poęcie dzewa AVL.

Definicja w linii 19 pliku DrzewoAVL.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 template<class typ> DrzewoAVL< typ >::DrzewoAVL ( ) [inline]

Konstrunkto bezargumentowy zeruje liczbę węzłów i ustawia Korzen na NULL.

Definicja w linii 243 pliku DrzewoAVL.hh.

4.2.2.2 template<class typ> DrzewoAVL< typ >::~DrzewoAVL ( ) [inline]

Destruktor zwalnia pamięć usuwając wszystkie węzły drzewa.

Definicja w linii 254 pliku DrzewoAVL.hh.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 template<class typ> void DrzewoAVL< typ >::Czysc (Wezel< typ > * *wezel*) [inline], [private]

Usuwa wszystkie węzły i zwalnia po nich pamięć leżące poniżej węzła podanego w argumecie (z nim włącznie)

Parametry

wezel	- węzeł od którego zaczyna się czyszczenie

Definicja w linii 227 pliku DrzewoAVL.hh.

4.2.3.2 template<class typ> void DrzewoAVL< typ >::CzyscDrzewo ( ) [inline]

Usuwa wszystkie węzły drzewa, walnia po nich pamięć, następnie ustawia korzeń na NULL i Liczbę węzłów na 0.

Definicja w linii 454 pliku DrzewoAVL.hh.

4.2.3.3 template<class typ> WezelAVL<typ>* DrzewoAVL< typ >::FindMin (WezelAVL< typ > * *poszukiwacz*) [inline], [private]

Szuka węzła o nejmniejeszej wartości, począwszy od węzła podanego w arunemcie.

Parametry

in	*poszukiwacz*	- węzeł startowy poszukwiań

Zwracane wartości

-	wskaźnik na węzeł przechowujący najmniejszą wartość

Definicja w linii 44 pliku DrzewoAVL.hh.

4.2.3.4 template<class typ> WezelAVL<typ>* DrzewoAVL< typ >::FindSuccessor (WezelAVL< typ > * *poszukiwacz*) [inline], [private]

Szuka poprzednika węzła podanego w artumencje wywołania.

Parametry

in	*poszukiwacz*	- węzeł którego poprzednik ma zostać znaleziony

Zwracane wartości

wskaźnik	na węzeł będący poprzednikiem

Definicja w linii 58 pliku DrzewoAVL.hh.

4.2.3.5 template<class typ> void DrzewoAVL< typ >::Insert (typ *wartosc*) [inline], [virtual]

Dodaje nową daną (węzeł) do struktury drzewa.

Parametry

in	*wartosc*	- dana do dodania

Implementuje IDrzew< typ >.

Definicja w linii 265 pliku DrzewoAVL.hh.

4.2.3.6 template<class typ> void DrzewoAVL< typ >::LL (WezelAVL< typ > * *A*) [inline], [private]

Metoda wykonuje rotację LL, gdzie A jest węzłem głównym rotacji.

Parametry

in	*A*	- wskaźnik do węzła głównego rotacji

Definicja w linii 111 pliku DrzewoAVL.hh.

4.2.3.7 template<class typ> void DrzewoAVL< typ >::LR (WezelAVL< typ > * *A*) [inline], [private]

Metoda wykonuje rotację LR, gdzie A jest węzłem głównym rotacji.

Parametry

in	*A*	- wskaźnik do węzła głównego rotacji

Definicja w linii 185 pliku DrzewoAVL.hh.

4.2.3.8 template<class typ> void DrzewoAVL< typ >::Remove (typ *wartosc*) [inline], [virtual]

Usuwa węzeł przechowujący daną wartość.

Parametry

in	*wartosc*	- przechowywana wartość do usunięcia z drzewa

Implementuje IDrzew< typ >.

Definicja w linii 342 pliku DrzewoAVL.hh.

4.2.3.9 template<class typ> void DrzewoAVL< typ >::RL (WezelAVL< typ > * *A*) [inline], [private]

Metoda wykonuje rotację RL, gdzie A jest węzłem głównym rotacji.

Parametry

in	*A*	- wskaźnik do węzła głównego rotacji

Definicja w linii 143 pliku DrzewoAVL.hh.

4.2.3.10 template<class typ> void DrzewoAVL< typ >::RR (WezelAVL< typ > * *A*) [inline], [private]

Metoda wykonuje rotację RR, gdzie A jest węzłem głównym rotacji.

Parametry

in	*A*	- wskaźnik do węzła głównego rotacji

Definicja w linii 77 pliku DrzewoAVL.hh.

4.2.3.11 template<class typ> WezelAVL<typ>* DrzewoAVL< typ >::Search (typ *wartosc*) [inline]

Wyszukuje i zwraca daną wartość z drzewa. W przypadku braku jej w drzewie i zwraca NULL

Parametry

in	*wartosc*	- wartość do znalezienia w drzewie

Zwracane wartości

-	odnaleziona wartość / NULL gdy brak wartości

Definicja w linii 436 pliku DrzewoAVL.hh.

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 template<class typ> WezelAVL<typ>* DrzewoAVL< typ >::Korzen [private]

Wskaźnik na węzeł będący korzeniem drzewa binarnego.

Definicja w linii 26 pliku DrzewoAVL.hh.

4.2.4.2 template<class typ> int DrzewoAVL< typ >::LiczbaWezlow [private]

Ilość węzłów w drzewie.

Definicja w linii 33 pliku DrzewoAVL.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

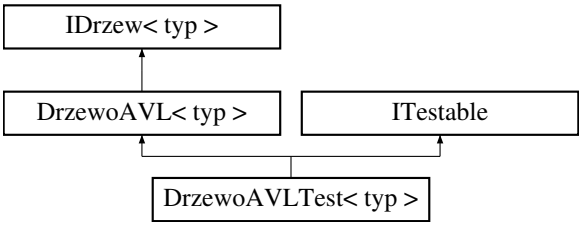
- DrzewoAVL.hh

4.3 Dokumentacja szablonu klasy DrzewoAVLTest< typ >

Testowalne Drzewo AVL.

`#include <DrzewoAVLTest.hh>`

Diagram dziedziczenia dla DrzewoAVLTest< typ >



Metody publiczne

- void WczytajDane (std::string const nazwaPliku, unsigned int n)
  Wczytanie danych z pliku.
- void Start (const unsigned int k, std::string const nazwaPliku)
  Wykonanie części obliczeniowej programu.

- void Zwolnij ()

  Zwalnia pamięć po teście.

4.3.1 Opis szczegółowy

template<class typ>class DrzewoAVLTest< typ >

Plik zawiera klasę modelującą pojęcie drzewa AVL z zaimplementowanymi metodami umożliwiajcymi jego testowanie.

The DrzewoAVLTest class

Klasa modelująca drzewo AVL z implemetacją metod niezbędncyh do testowania.

Definicja w linii 22 pliku DrzewoAVLTest.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 template<class typ > void DrzewoAVLTest< typ >::Start (const unsigned int *k,* std::string const *nazwaPliku*) [inline], [virtual]

Metoda w której implementowana jest część obliczeniowa programu, której czas wykonania zostanie zmierzony.

Parametry

in	*k*	- ilość elementów dla których mają zostać wykonane obliczenia.
in	*nazwaPliku*	- nazwa pliku z danymi

Implementuje ITestable.

Definicja w linii 57 pliku DrzewoAVLTest.hh.

4.3.2.2 template<class typ > void DrzewoAVLTest< typ >::WczytajDane (std::string const *nazwaPliku,* unsigned int *n*) [inline], [virtual]

Wczytuje zadaną ilość danych do przetworzenia z pliku o zadanej nazwie.

Parametry

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania

Implementuje ITestable.

Definicja w linii 36 pliku DrzewoAVLTest.hh.

4.3.2.3 template<class typ > void DrzewoAVLTest< typ >::Zwolnij ( ) [inline], [virtual]

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

Implementuje ITestable.

Definicja w linii 67 pliku DrzewoAVLTest.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

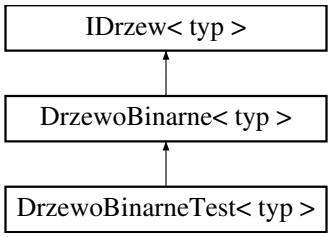
- DrzewoAVLTest.hh

4.4 Dokumentacja szablonu klasy DrzewoBinarne< typ >

DrzewoBinarne.

`#include <DrzewoBinarne.hh>`

Diagram dziedziczenia dla DrzewoBinarne< typ >



Metody publiczne

- DrzewoBinarne ()
  Konstruktor bezarumentowy.
- ~DrzewoBinarne ()
  Destruktor.
- void Insert (const typ wartosc)
  Dodaje element.
- void Remove (typ wartosc)
  Remove.
- Wezel< typ > * Search (typ wartosc)
  Search.
- void CzyscDrzewo ()
  CzyscDrzewo.

Metody prywatne

- Wezel< typ > * FindMin (Wezel< typ > *poszukiwacz)
  FindMin.
- Wezel< typ > * FindSuccessor (Wezel< typ > *poszukiwacz)
  FindSuccessor.
- void Czysc (Wezel< typ > *wezel)
  Czyści drzewo.

Atrybuty prywatne

- Wezel< typ > * Korzen
  Korzeń drzewa.
- int LiczbaWezlow
  Liczba Węzłów.

4.4.1 Opis szczegółowy

template<class typ>class DrzewoBinarne< typ >

Plik zawiera definicje klasy DrzewoBinarne.

Klasa DrzewoBinarne

Klasa modeluje Drzewo Binarne.

Definicja w linii 19 pliku DrzewoBinarne.hh.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 template<class typ> DrzewoBinarne< typ >::DrzewoBinarne ( ) [inline]

Konstrunkto bezargumentowy zeruje liczbę węzłów i ustawia Korzen na NULL.

Definicja w linii 94 pliku DrzewoBinarne.hh.

4.4.2.2 template<class typ> DrzewoBinarne< typ >::~DrzewoBinarne ( ) [inline]

Destruktor zwalnia pamięć usuwając wszystkie węzły drzewa.

Definicja w linii 105 pliku DrzewoBinarne.hh.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 template<class typ> void DrzewoBinarne< typ >::Czysc (Wezel< typ > * *wezel*) [inline], [private]

Usuwa wszystkie węzły i zwalnia po nich pamięć leżące poniżej węzła węzła podanego w argumecie (z nim włącznie)

Parametry

wezel	- węzeł od którego zaczyna się czyszczenie

Definicja w linii 78 pliku DrzewoBinarne.hh.

4.4.3.2 template<class typ> void DrzewoBinarne< typ >::CzyscDrzewo ( ) [inline]

Usuwa wszystkie węzły drzewa, walnia po nich pamięć, następnie ustawia korzeń na NULL i Liczbę węzłów na 0.

Definicja w linii 214 pliku DrzewoBinarne.hh.

4.4.3.3 template<class typ> Wezel<typ>* DrzewoBinarne< typ >::FindMin (Wezel< typ > * *poszukiwacz*) [inline], [private]

Szuka węzła o nejmniejeszej wartości, począwszy od węzła podanego w arunemcie.

Parametry

in	*poszukiwacz*	- węzeł startowy poszukwiań

Zwracane wartości

-	wskaźnik na węzeł przechowujący najmniejszą wartość

Definicja w linii 44 pliku DrzewoBinarne.hh.

4.4.3.4 template<class typ> Wezel<typ>* DrzewoBinarne< typ >::FindSuccessor (Wezel< typ > * *poszukiwacz*) [inline], [private]

Szuka poprzednika węzła podanego w artumencje wywołania.

Parametry

in	*poszukiwacz*	- węzeł którego poprzednik ma zostać znaleziony

Zwracane wartości

wskaźnik	na węzeł będący poprzednikiem

Definicja w linii 58 pliku DrzewoBinarne.hh.

4.4.3.5 template<class typ> void DrzewoBinarne< typ >::Insert (const typ *wartosc*) [inline], [virtual]

Dodaje element do drzewa binarnego

Parametry

in	*wartosc*	- wartość do umieszczenia w drzewie.

Implementuje IDrzew< typ >.

Definicja w linii 116 pliku DrzewoBinarne.hh.

4.4.3.6 template<class typ> void DrzewoBinarne< typ >::Remove (typ *wartosc*) [inline], [virtual]

Usuwa węzeł przechowujący daną wartość.

Parametry

in	*wartosc*	- przechowywana wartość do usunięcia z drzewa

Implementuje IDrzew< typ >.

Definicja w linii 153 pliku DrzewoBinarne.hh.

4.4.3.7 template<class typ> Wezel<typ>* DrzewoBinarne< typ >::Search (typ *wartosc*) [inline]

Wyszukuje i zwraca daną wartość z drzewa. W przypadku braku jej w drzewie wyświetla stosowny błąd i zwraca NULL / 0

Parametry

in	*wartosc*	- wartość do znalezienia w drzewie

Zwracane wartości

-	odnaleziona wartość

Definicja w linii 196 pliku DrzewoBinarne.hh.

4.4.4 Dokumentacja atrybutów składowych

4.4.4.1 template<class typ> Wezel<typ>* DrzewoBinarne< typ >::Korzen [private]

Wskaźnik na węzeł będący korzeniem drzewa binarnego.

Definicja w linii 26 pliku DrzewoBinarne.hh.

4.4.4.2 template<class typ> int DrzewoBinarne< typ >::LiczbaWezlow [private]

Ilość węzłów w drzewie.

Definicja w linii 33 pliku DrzewoBinarne.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

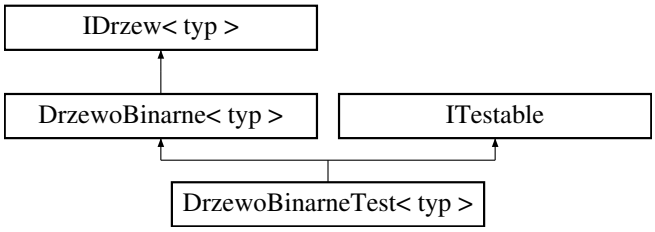
- DrzewoBinarne.hh

4.5 Dokumentacja szablonu klasy DrzewoBinarneTest< typ >

Drzewo binarne testowalne.

`#include <DrzewoBinarneTest.hh>`

Diagram dziedziczenia dla DrzewoBinarneTest< typ >



Metody publiczne

- void WczytajDane (std::string const nazwaPliku, unsigned int n)

  Wczytanie danych z pliku.
- void Start (const unsigned int k, std::string const nazwaPliku)

  Wykonanie części obliczeniowej programu.
- void Zwolnij ()

  Zwalnia pamięć po teście.

4.5.1 Opis szczegółowy

template<class typ>class DrzewoBinarneTest< typ >

Plik zawiera implemementację testowalnego drzewa binarnego.

The DrzewoBinaneTest class

Drzewo binarne z zaimplemetowanym interfejejsem do testów.

Definicja w linii 19 pliku DrzewoBinarneTest.hh.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 template<class typ> void DrzewoBinarneTest< typ >::Start (const unsigned int *k*, std::string const *nazwaPliku*) [inline], [virtual]

Metoda w której implementowana jest część obliczeniowa programu, której czas wykonania zostanie zmierzony.

Parametry

in	*k*	- ilość elementów dla których mają zostać wykonane obliczenia.
in	*nazwaPliku*	- nazwa pliku z danymi

Implementuje ITestable.

Definicja w linii 54 pliku DrzewoBinarneTest.hh.

4.5.2.2 template<class typ> void DrzewoBinarneTest< typ >::WczytajDane (std::string const *nazwaPliku*, unsigned int *n*) [inline], [virtual]

Wczytuje zadaną ilość danych do przetworzenia z pliku o zadanej nazwie.

Parametry

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania

Implementuje ITestable.

Definicja w linii 33 pliku DrzewoBinarneTest.hh.

4.5.2.3 template<class typ> void DrzewoBinarneTest< typ >::Zwolnij ( ) [inline], [virtual]

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

Implementuje ITestable.

Definicja w linii 64 pliku DrzewoBinarneTest.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

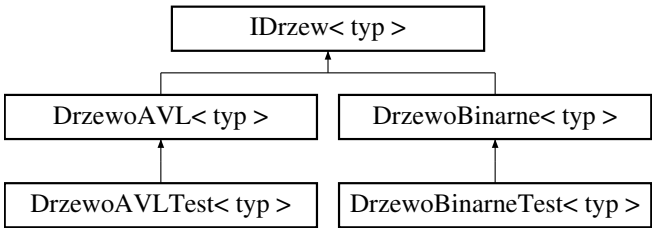
- DrzewoBinarneTest.hh

4.6 Dokumentacja szablonu klasy IDrzew< typ >

Definicja IDrzew.

`#include <IDrzew.hh>`

Diagram dziedziczenia dla IDrzew< typ >



Metody publiczne

- virtual void Insert (typ wartosc)=0

  Insert.
- virtual void Remove (typ wartosc)=0

  Remove.

4.6.1 Opis szczegółowy

template<class typ>class IDrzew< typ >

Plik zawiera definicję interfejsu IDrzew

The IDrzewclass

Klasa czysto abstrancyjna modelująca interfejs użytkownika dla drzew.

Definicja w linii 17 pliku IDrzew.hh.

4.6.2 Dokumentacja funkcji składowych

4.6.2.1 template<class typ > virtual void IDrzew< typ >::Insert (typ *wartosc*) [pure virtual]

Dodaje nową daną (węzeł) do struktury drzewa.

Parametry

in	*wartosc*	- dana do dodania

Implementowany w DrzewoAVL< typ > i DrzewoBinarne< typ >.

4.6.2.2 template<class typ > virtual void IDrzew< typ >::Remove (typ *wartosc*) [pure virtual]

Usuwa węzeł przechowujący daną wartość.

Parametry

in	*wartosc*	- przechowywana wartość do usunięcia z drzewa

Implementowany w DrzewoAVL< typ > i DrzewoBinarne< typ >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

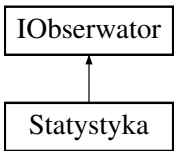
- IDrzew.hh

4.7 Dokumentacja klasy IObserwator

Klasa IObserwator.

`#include <IObserwator.hh>`

Diagram dziedziczenia dla IObserwator



Metody publiczne

- virtual void Aktualizuj ()=0

  Aktualizuj.

4.7.1 Opis szczegółowy

Plik zawira definicję klasy IObsereator.

The IObserwator class

Klasa modeluje interfejs objektu będącego obserwatorem.

Definicja w linii 17 pliku IObserwator.hh.

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 virtual void IObserwator::Aktualizuj ( ) [pure virtual]

Aktualizuje dane na podstawie wydarzenie w objekcie obserowanym.

Implementowany w Statystyka.

Dokumentacja dla tej klasy została wygenerowana z pliku:

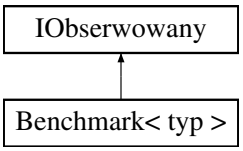
- IObserwator.hh

4.8 Dokumentacja klasy IObserwowany

Interfejs obserwowanego.

`#include <IObserwowany.hh>`

Diagram dziedziczenia dla IObserwowany



Metody publiczne

- virtual void DodajObserwatora (IObserwator ∗nowyObserwator)=0

  Dodaje Obserwatora.
- virtual void UsunObserwatora (IObserwator ∗obserwator)=0

  Usuwa Obserwatora.
- virtual void PowiadomObserwatorow ()=0

  Powiadamia Obserwatorów.

4.8.1 Opis szczegółowy

W pliku zawarta jest definicja interfejsu obserwowanego

The IObserwowany class

Klasa czysto wirtualna modelująca interfejs objektu obserwowanego.

Definicja w linii 19 pliku IObserwowany.hh.

4.8.2 Dokumentacja funkcji składowych

4.8.2.1 virtual void IObserwowany::DodajObserwatora (IObserwator ∗ *nowyObserwator*) [pure virtual]

Dodaje nowego obserwatora do listy oserwatorów danego objektu.

Parametry

in	*nowyObserwator*	- wkaźnik na dodawanego obserwatora

Implementowany w Benchmark< typ >.

4.8.2.2 virtual void IObserwowany::PowiadomObserwatorow ( ) [pure virtual]

Powiadamia obseratorów o wydarzeniu.

Implementowany w Benchmark< typ >.

4.8.2.3 virtual void IObserwowany::UsunObserwatora (IObserwator ∗ *obserwator*) [pure virtual]

Usuwa danego obserwatora z listy obserwatorów danego objektu.

Parametry

in	*obserwator*	- obserwator do usunięcia z listy

Implementowany w Benchmark< typ >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

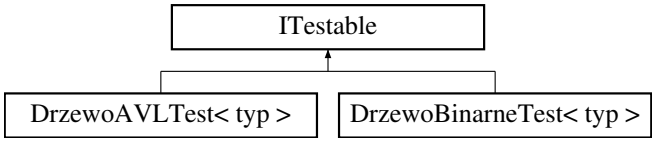
- IObserwowany.hh

4.9 Dokumentacja klasy ITestable

Modeluje interfejs programu.

`#include <ITestable.hh>`

Diagram dziedziczenia dla ITestable



Metody publiczne

- virtual void WczytajDane (std::string const nazwaPliku, unsigned int n)=0

  Wczytanie danych z pliku.
- virtual void Start (const unsigned int k, std::string const nazwaPliku)=0

  Wykonanie części obliczeniowej programu.
- virtual void Zwolnij ()=0

  Zwalnia pamięć po teście.

4.9.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanch w ramach kursu.

Definicja w linii 24 pliku ITestable.hh.

4.9.2 Dokumentacja funkcji składowych

4.9.2.1 virtual void ITestable::Start (const unsigned int *k*, std::string const *nazwaPliku*) [pure virtual]

Metoda w której implementowana jest część obliczeniowa programu, której czas wykonania zostanie zmierzony.

Parametry

in	*k*	- ilość elementów dla których mają zostać wykonane obliczenia.

Implementowany w DrzewoAVLTest< typ > i DrzewoBinarneTest< typ >.

4.9.2.2 virtual void ITestable::WczytajDane (std::string const *nazwaPliku*, unsigned int *n*) [pure virtual]

Wczytuje zadaną ilość danych do przetworzenia z pliku o zadanej nazwie.

Parametry

in	*nazwaPliku*	- nazwa pliku z danymi
in	*n*	- ilość danych do wczytania

Implementowany w DrzewoAVLTest< typ > i DrzewoBinarneTest< typ >.

4.9.2.3 virtual void ITestable::Zwolnij ( ) [pure virtual]

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

Implementowany w DrzewoAVLTest< typ > i DrzewoBinarneTest< typ >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

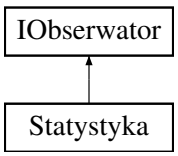
- ITestable.hh

4.10 Dokumentacja klasy Statystyka

Modeluje pojęcie statystyki.

`#include <Statystyka.hh>`

Diagram dziedziczenia dla Statystyka



Metody publiczne

- Statystyka (const unsigned int iloscProb, unsigned int ∗proby, const unsigned int ilePowtorzen)

  Konstruktor z dwoma pramametrami.
- ∼Statystyka ()

  Destruktor - zwaknia pamięć
- void ZapiszStaty (std::string nazwaPliku) const

  Zapisuje statysykę do pliku.
- void Aktualizuj ()

  Aktualizuj.

Atrybuty prywatne

- unsigned int IleProb

  Ilość prób.
- unsigned int ∗ Proba

  Tablica z rozmiarami prób.
- double ∗ Czas

  Średni czas wykonania danej próby.
- double SumaCzasuProby

  Suma Czasu Proby.
- unsigned int IloscPowtorzen

  Ilość Powtórzeń
- unsigned int LicznikPowtorzen

Licznik Powtórzeń

- unsigned int LicznikProb

  Licznik Prób.

- Stoper * MojStoper

  Stoper.

4.10.1 Opis szczegółowy

Modeluje pojęcie statystyki, czyli średnich czasów wykonania metody dla różnyuch wielkości prób.

Definicja w linii 27 pliku Statystyka.hh.

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 Statystyka::Statystyka (const unsigned int *iloscProb,* unsigned int * *proby,* const unsigned int *ilePowtorzen*)

Konstruktor z dwoma paramatremi tworzy dynamiczne tablice przechowujące statystykę oraz wypełnia rozmiary prób.

Parametry

in	*iloscProb*	- liczbosc prob w ksperymencie
in	*proby*	- tablica z licznościami prób.

Definicja w linii 12 pliku Statystyka.cpp.

4.10.2.2 Statystyka::~Statystyka ( ) [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablicy przechowujące statystykę.

Definicja w linii 108 pliku Statystyka.hh.

4.10.3 Dokumentacja funkcji składowych

4.10.3.1 void Statystyka::Aktualizuj ( ) [virtual]

Aktualizuje pozyskiwane dane dotyczące wyników testu: Jeżeli stoper nie odlicza to uruchamia odliczanie, Jeżeli stper odlicza to go zatrzymuje i sumuje czasy powtórzeń. Gdy nasąpi wykonanie wszystkich pomiarów w próbie to uzupełnia talicę przechowywujacą średnie czasy każdej próby.

Implementuje IObserwator.

Definicja w linii 44 pliku Statystyka.cpp.

4.10.3.2 void Statystyka::ZapiszStaty (std::string *nazwaPliku*) const

Zapisuje statystystykę do pliku o nazwie podanej w argumencie. Plik zapisany zostaje w sposób, gdzie każda nowa linia wygląda następująco: RozmiarPróby,ŚredniCzas czas wyrażony jest w ms.

Parametry

in	*nazwaPliku*	- nazwa pliku do którego ma zostać zapisanaza statystyka

Definicja w linii 25 pliku Statystyka.cpp.

4.10.4 Dokumentacja atrybutów składowych

4.10.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 51 pliku Statystyka.hh.

4.10.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 35 pliku Statystyka.hh.

4.10.4.3 unsigned int Statystyka::IloscPowtorzen [private]

Przechowuje ilość wykonywanych powtórzeń pojedyńczego testu.

Definicja w linii 65 pliku Statystyka.hh.

4.10.4.4 unsigned int Statystyka::LicznikPowtorzen [private]

Zlicza ilosć wykonanych powtórzeń w danej próbie.

Definicja w linii 72 pliku Statystyka.hh.

4.10.4.5 unsigned int Statystyka::LicznikProb [private]

Zlicza ilosć prób wykonanych prób.

Definicja w linii 79 pliku Statystyka.hh.

4.10.4.6 Stoper* Statystyka::MojStoper [private]

Stoper wykorzystywany do pomiaru czasu.

Definicja w linii 86 pliku Statystyka.hh.

4.10.4.7 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 43 pliku Statystyka.hh.

4.10.4.8 double Statystyka::SumaCzasuProby [private]

Przechowuje sumę czasów pojedyńczych powtórzeń z danej próby.

Definicja w linii 58 pliku Statystyka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Statystyka.hh
- Statystyka.cpp

4.11 Dokumentacja klasy Stoper

Klasa Stoper.

`#include <Stoper.hh>`

Metody publiczne

- Stoper ()
  Stoper.
- void Start ()
  Start.
- void Stop ()

Stop.

- void Reset ()

  Reset.

- double DajPomiar () const

  Pomiar.

- bool CzyOdmierza () const

  Czy Odmierza.

Atrybuty prywatne

- double CzasPoczatkowy

  Czas Początkowy.

- double CzasKoncowy

  Czas Końcowy.

- bool CzyLiczy

  Czy Liczy.

4.11.1 Opis szczegółowy

Plik zawiera definicję klasy Stoper.

The Stoper class

Klasa modeluje stoper niezbędny do odliczania czasu.

Definicja w linii 20 pliku Stoper.hh.

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 Stoper::Stoper ( )

Kontruktor bezarumentowy zeruje czasy i ustawia wartość pola CzyLiczy na false.

Definicja w linii 3 pliku Stoper.cpp.

4.11.3 Dokumentacja funkcji składowych

4.11.3.1 bool Stoper::CzyOdmierza ( ) const

Informuje czy stoper aktualinie liczy czy nie.

Zwracane wartości

true	- gdy odlicza
false	- gdy nie odlicza

Definicja w linii 29 pliku Stoper.cpp.

4.11.3.2 double Stoper::DajPomiar ( ) const

Wyłuskuje czas pomiaru w ms.

Zwracane wartości

zwrca	czas pomiaru wyrażon w ms

Definicja w linii 25 pliku Stoper.cpp.

4.11.3.3 void Stoper::Reset ( )

Resetuje stoper.

Definicja w linii 19 pliku Stoper.cpp.

4.11.3.4 void Stoper::Start ( )

Uruchamia odliczanie czasu.

Definicja w linii 9 pliku Stoper.cpp.

4.11.3.5 void Stoper::Stop ( )

Zatrzymuje odliczanie czasu.

Definicja w linii 14 pliku Stoper.cpp.

4.11.4 Dokumentacja atrybutów składowych

4.11.4.1 double Stoper::CzasKoncowy [private]

Czas w którym odliczanie czasu zostało zatrzymane.

Definicja w linii 34 pliku Stoper.hh.

4.11.4.2 double Stoper::CzasPoczatkowy [private]

Czas w którym stoper zaczął odliczać.

Definicja w linii 27 pliku Stoper.hh.

4.11.4.3 bool Stoper::CzyLiczy [private]

Zmienna przechowuje wartośc true gdy stoper aktualnie odlicza czas, lub false gdy jest zatrzymany.

Definicja w linii 42 pliku Stoper.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Stoper.hh
- Stoper.cpp

4.12 Dokumentacja szablonu struktury Wezel< typ >

Klasa węzeł

`#include <Wezel.hh>`

Metody publiczne

- Wezel ()
- ~Wezel ()

Atrybuty publiczne

- typ Dana

Przechowywana wartość

- Wezel * Rodzic

  Rodzic danego węzła.

- Wezel * Lewy

  Lewy potomek.

- Wezel * Prawy

  Prawy potomek.

4.12.1 Opis szczegółowy

template<class typ>struct Wezel< typ >

Plik zawiera definicję strukntury Węzeł.

Struktura Węzeł

Struktura modeluje pojęcie węzła - elementu drzewa, na który składa się wartość, rodzic, lewy potomek i prawy potomek.

Definicja w linii 22 pliku Wezel.hh.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 template<class typ> Wezel< typ >::Wezel ( ) [inline]

Definicja w linii 54 pliku Wezel.hh.

4.12.2.2 template<class typ> Wezel< typ >::~Wezel ( ) [inline]

Definicja w linii 60 pliku Wezel.hh.

4.12.3 Dokumentacja atrybutów składowych

4.12.3.1 template<class typ> typ Wezel< typ >::Dana

Pole przechowuje wartość elementu znajdującego się w danym węźle.

Definicja w linii 29 pliku Wezel.hh.

4.12.3.2 template<class typ> Wezel* Wezel< typ >::Lewy

Wskaźnik na lewego potomka danego węzła.

Definicja w linii 43 pliku Wezel.hh.

4.12.3.3 template<class typ> Wezel* Wezel< typ >::Prawy

Wskaźnik na prawego potomnka danego węzła.

Definicja w linii 50 pliku Wezel.hh.

4.12.3.4 template<class typ> Wezel* Wezel< typ >::Rodzic

Wskaźnik na rodzica danego węzła.

Definicja w linii 36 pliku Wezel.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Wezel.hh

4.13 Dokumentacja szablonu struktury WezelAVL< typ >

Węzeł drzewa AVL.

`#include <WezelAVL.hh>`

Metody publiczne

- WezelAVL ()

Atrybuty publiczne

- typ Dana
  Przechowywana wartość
- WezelAVL * Rodzic
  Rodzic danego węzła.
- WezelAVL * Lewy
  Lewy potomek.
- WezelAVL * Prawy
  Prawy potomek.
- int WspRownowagi

4.13.1 Opis szczegółowy

template<class typ>struct WezelAVL< typ >

Plik zawiera definicję węzła wykorzytywanego w drzewie AVL.

Węzeł drzewa AVL

Struktura nędąca reprezentacją pojednyńczego węzła w drzewie AVL.

Definicja w linii 17 pliku WezelAVL.hh.

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 template<class typ> WezelAVL< typ >::WezelAVL ( ) [inline]

Definicja w linii 49 pliku WezelAVL.hh.

4.13.3 Dokumentacja atrybutów składowych

4.13.3.1 template<class typ> typ WezelAVL< typ >::Dana

Pole przechowuje wartość elementu znajdującego się w danym węźle.

Definicja w linii 24 pliku WezelAVL.hh.

4.13.3.2 template<class typ> WezelAVL* WezelAVL< typ >::Lewy

Wskaźnik na lewego potomka danego węzła.

Definicja w linii 38 pliku WezelAVL.hh.

4.13.3.3 template<class typ> WezelAVL* WezelAVL< typ >::Prawy

Wskaźnik na prawego potomnka danego węzła.

Definicja w linii 45 pliku WezelAVL.hh.

4.13.3.4 template<class typ> WezelAVL* WezelAVL< typ >::Rodzic

Wskaźnik na rodzica danego węzła.

Definicja w linii 31 pliku WezelAVL.hh.

4.13.3.5 template<class typ> int WezelAVL< typ >::WspRownowagi

Definicja w linii 47 pliku WezelAVL.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- WezelAVL.hh

5 Dokumentacja plików

5.1 Dokumentacja pliku Benchmark.hh

Definicja klasy Benchmark.

```
#include <ctime>
#include "Statystyka.hh"
#include "IObserwowany.hh"
#include <list>
#include "ITestable.hh"
```

Komponenty

- class Benchmark< typ >

  Modeluje pojęcie Benchmarku.

5.1.1 Opis szczegółowy

Plik zawiera definicję klasy Benchmark wraz z definicją jej metod.

Definicja w pliku Benchmark.hh.

5.2 Dokumentacja pliku DrzewoAVL.hh

```
#include "IDrzew.hh"
#include "WezelAVL.hh"
```

Komponenty

- class DrzewoAVL< typ >

  Definicja drzewa AVL.

5.3 Dokumentacja pliku DrzewoAVLTest.hh

```
#include "DrzewoAVL.hh"
#include "ITestable.hh"
```

Komponenty

- class DrzewoAVLTest< typ >

  Testowalne Drzewo AVL.

5.4 Dokumentacja pliku DrzewoBinarne.hh

```
#include "IDrzew.hh"
```

Komponenty

- class DrzewoBinarne< typ >

  DrzewoBinarne.

5.5 Dokumentacja pliku DrzewoBinarneTest.hh

```
#include "DrzewoBinarne.hh"
#include "Pliki.hh"
```

Komponenty

- class DrzewoBinarneTest< typ >

  Drzewo binarne testowalne.

5.6 Dokumentacja pliku IDrzew.hh

```
#include "Wezel.hh"
```

Komponenty

- class IDrzew< typ >

  Definicja IDrzew.

5.7 Dokumentacja pliku IObserwator.hh

Komponenty

- class IObserwator

  Klasa IObserwator.

5.8 Dokumentacja pliku IObserwowany.hh

```
#include "IObserwator.hh"
```

Komponenty

- class IObserwowany

  Interfejs obserwowanego.

Definicje

- #define IOBSERWOWANY_HH

5.8.1 Dokumentacja definicji

5.8.1.1 #define IOBSERWOWANY_HH

Definicja w linii 2 pliku IObserwowany.hh.

5.9 Dokumentacja pliku ITestable.hh

Definicja klasy ITestable.

```
#include <iostream>
```

Komponenty

- class ITestable

  Modeluje interfejs programu.

5.9.1 Opis szczegółowy

Plik zawiera definicję abstrakcyjnej klasy ITestable, która tworzy interfejs dla programów implementowanych podczas zajęć laboratoryjnych z PAMSI.

Definicja w pliku ITestable.hh.

5.10 Dokumentacja pliku main.cpp

Moduł główny programu.

```
#include "../inc/Statystyka.hh"
#include "../inc/Benchmark.hh"
#include "../inc/Pliki.hh"
#include "../inc/DrzewoBinarneTest.hh"
#include "../inc/DrzewoAVLTest.hh"
```

Funkcje

- int main (int argc, char ∗argv[])

Zmienne

- const int ILOSC_POWTORZEN = 50

  Ilośc powtórzeń danej próby.

- const int ILOSC_PROB = 11

  Ilość prób.
- const std::string NAZWA_PLIKU_Z_DANYMI = "dane.dat"

5.10.1 Opis szczegółowy

Program wkonuje serię 50 pomiarów czasu wykonania metody start dla różncyh wielkości problemu obliczeniowego. OBSŁUGA PROGRAMU: Aby wywołać program należy w lini poleceń wywołać jego nazę np: "./a.out"

Definicja w pliku main.cpp.

5.10.2 Dokumentacja funkcji

5.10.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Definicja w linii 37 pliku main.cpp.

5.10.3 Dokumentacja zmiennych

5.10.3.1 const int ILOSC_POWTORZEN = 50

Ilośc powtórzeń danej próby

Definicja w linii 25 pliku main.cpp.

5.10.3.2 const int ILOSC_PROB = 11

Ilość prób = ilość rozmiarów prób

Definicja w linii 33 pliku main.cpp.

5.10.3.3 const std::string NAZWA_PLIKU_Z_DANYMI = "dane.dat"

Definicja w linii 35 pliku main.cpp.

5.11 Dokumentacja pliku Pliki.cpp

Definicje funkcji obslugi plikow.

```
#include "../inc/Pliki.hh"
```

Funkcje

- void OtworzPlikIn (const char ∗nazwaPliku, std::fstream &plik)

  Otwiera plik do odczytu.
- void OtworzPlikOut (const char ∗nazwaPliku, std::fstream &plik)

  Otwiera plik do zapisu czyszcąc jego zawartość
- void LosujIntRandDoPliku (const unsigned int n, const unsigned int zakres)

  Zapisuje n losowych liczb(int) do pliku.
- void LosujIntRosnacoDoPliku (const unsigned int n, const unsigned int zakres)

  Zapisuje n losowych liczb(int) do pliku.

5.11.1 Opis szczegółowy

Plik zawiera definicje funkcji zwiazanych z obsluga plikow.

Definicja w pliku Pliki.cpp.

5.11.2 Dokumentacja funkcji

5.11.2.1 void LosujIntRandDoPliku (const unsigned int *n*, const unsigned int *zakres*)

Losuje n liczb z zakresu od 1 do podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parametry

in	*n*	- ilość liczb do zapisania
in	*zakres*	- górny zakres wartości liczb

Definicja w linii 27 pliku Pliki.cpp.

5.11.2.2 void LosujIntRosnacoDoPliku (const unsigned int *n*, const unsigned int *zakres*)

Losuje n liczb z zakresu od 1 do podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parametry

in	*n*	- ilość liczb do zapisania
in	*zakres*	- górny zakres wartości liczb

Definicja w linii 44 pliku Pliki.cpp.

5.11.2.3 void OtworzPlikIn (const char * *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.11.2.4 void OtworzPlikOut (const char * *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.12 Dokumentacja pliku Pliki.hh

Funkcje obslugi plikow.

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

Funkcje

- void OtworzPlikIn (const char ∗nazwaPliku, std::fstream &plik)

  Otwiera plik do odczytu.
- void OtworzPlikOut (const char ∗nazwaPliku, std::fstream &plik)

  Otwiera plik do zapisu czyszcąc jego zawartość
- void LosujIntRandDoPliku (const unsigned int n, const unsigned int zakres)

  Zapisuje n losowych liczb(int) do pliku.
- void LosujIntRosnacoDoPliku (const unsigned int n, const unsigned int zakres)

  Zapisuje n losowych liczb(int) do pliku.

5.12.1 Opis szczegółowy

Plik zawiera deklaracje funkcji zwiazanych z obsuga plikow

Definicja w pliku Pliki.hh.

5.12.2 Dokumentacja funkcji

5.12.2.1 void LosujIntRandDoPliku (const unsigned int *n*, const unsigned int *zakres*)

Losuje n liczb z zakresu od 1 do podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parametry

in	*n*	- ilość liczb do zapisania
in	*zakres*	- górny zakres wartości liczb

Definicja w linii 27 pliku Pliki.cpp.

5.12.2.2 void LosujIntRosnacoDoPliku (const unsigned int *n*, const unsigned int *zakres*)

Losuje n liczb z zakresu od 1 do podonago przez użytwkonika następnie zapisuje wylosowane dane do pliku o nazwe "dane.dat"

Parametry

in	*n*	- ilość liczb do zapisania
in	*zakres*	- górny zakres wartości liczb

Definicja w linii 44 pliku Pliki.cpp.

5.12.2.3 void OtworzPlikIn (const char ∗ *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 11 pliku Pliki.cpp.

5.12.2.4 void OtworzPlikOut (const char ∗ *nazwaPliku*, std::fstream & *plik*)

Otwiera plik i sprawdza czy otwarcie sie powiodlo jezeli nie to koczy program

Parametry

in	*nazwaPliku*	- nazwa pliku ktory chcemy otworzyc
in	*plik*	- strumien powiazany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.13 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy Statystyka.

```
#include "../inc/Statystyka.hh"
```

5.13.1 Opis szczegółowy

Plik zawiera definicję metod klasy Statystyka.

Definicja w pliku Statystyka.cpp.

5.14 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy Statystyka.

```
#include <iostream>
#include "IObserwator.hh"
#include "Stoper.hh"
#include <fstream>
#include <cstdlib>
#include <string>
```

Komponenty

- class Statystyka

  Modeluje pojęcie statystyki.

5.14.1 Opis szczegółowy

Zawiera definicję klasy Statystyka

Definicja w pliku Statystyka.hh.

5.15 Dokumentacja pliku Stoper.cpp

```
#include "../inc/Stoper.hh"
```

5.16 Dokumentacja pliku Stoper.hh

```
#include <iostream>
#include <ctime>
```

Komponenty

- class Stoper

  Klasa Stoper.

5.17 Dokumentacja pliku Wezel.hh

```
#include <iostream>
```

Komponenty

- struct Wezel< typ >

  Klasa węzeł

5.18 Dokumentacja pliku WezelAVL.hh

Komponenty

- struct WezelAVL< typ >

  Węzeł drzewa AVL.

Skorowidz