

PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Śr, 20 maj 2015 15:47:24

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	2
2.1 Lista klas	2
3 Indeks plików	2
3.1 Lista plików	2
4 Dokumentacja klas	3
4.1 Dokumentacja szablonu klasy Benchmark< typ >	3
4.1.1 Opis szczegółowy	4
4.1.2 Dokumentacja konstruktora i destruktora	4
4.1.3 Dokumentacja funkcji składowych	4
4.1.4 Dokumentacja atrybutów składowych	5
4.2 Dokumentacja szablonu klasy DrzewoAVL< typ >	6
4.2.1 Opis szczegółowy	7
4.2.2 Dokumentacja konstruktora i destruktora	7
4.2.3 Dokumentacja funkcji składowych	7
4.2.4 Dokumentacja atrybutów składowych	9
4.3 Dokumentacja szablonu klasy DrzewoAVLTest< typ >	9
4.3.1 Opis szczegółowy	10
4.3.2 Dokumentacja funkcji składowych	10
4.4 Dokumentacja szablonu klasy DrzewoBinarne< typ >	10
4.4.1 Opis szczegółowy	11
4.4.2 Dokumentacja konstruktora i destruktora	12
4.4.3 Dokumentacja funkcji składowych	12
4.4.4 Dokumentacja atrybutów składowych	13
4.5 Dokumentacja szablonu klasy DrzewoBinarneTest< typ >	13
4.5.1 Opis szczegółowy	14
4.5.2 Dokumentacja funkcji składowych	14
4.6 Dokumentacja szablonu klasy IDrzew< typ >	15
4.6.1 Opis szczegółowy	15
4.6.2 Dokumentacja funkcji składowych	15
4.7 Dokumentacja klasy IObservator	17
4.7.1 Opis szczegółowy	17
4.7.2 Dokumentacja funkcji składowych	17
4.8 Dokumentacja klasy IObservowany	18
4.8.1 Opis szczegółowy	18

4.8.2	Dokumentacja funkcji składowych	18
4.9	Dokumentacja klasy ITestable	19
4.9.1	Opis szczegółowy	19
4.9.2	Dokumentacja funkcji składowych	19
4.10	Dokumentacja klasy Statystyka	20
4.10.1	Opis szczegółowy	21
4.10.2	Dokumentacja konstruktora i destruktora	21
4.10.3	Dokumentacja funkcji składowych	21
4.10.4	Dokumentacja atrybutów składowych	21
4.11	Dokumentacja klasy Stoper	22
4.11.1	Opis szczegółowy	23
4.11.2	Dokumentacja konstruktora i destruktora	23
4.11.3	Dokumentacja funkcji składowych	23
4.11.4	Dokumentacja atrybutów składowych	24
4.12	Dokumentacja szablonu struktury Wezel< typ >	24
4.12.1	Opis szczegółowy	25
4.12.2	Dokumentacja konstruktora i destruktora	25
4.12.3	Dokumentacja atrybutów składowych	25
4.13	Dokumentacja szablonu struktury WezelAVL< typ >	26
4.13.1	Opis szczegółowy	26
4.13.2	Dokumentacja konstruktora i destruktora	26
4.13.3	Dokumentacja atrybutów składowych	26
5	Dokumentacja plików	27
5.1	Dokumentacja pliku Benchmark.hh	27
5.1.1	Opis szczegółowy	27
5.2	Dokumentacja pliku DrzewoAVL.hh	27
5.3	Dokumentacja pliku DrzewoAVLTest.hh	28
5.4	Dokumentacja pliku DrzewoBinarne.hh	28
5.5	Dokumentacja pliku DrzewoBinarneTest.hh	28
5.6	Dokumentacja pliku IDrzew.hh	28
5.7	Dokumentacja pliku IObserwator.hh	28
5.8	Dokumentacja pliku IObserwowany.hh	29
5.8.1	Dokumentacja definicji	29
5.9	Dokumentacja pliku ITestable.hh	29
5.9.1	Opis szczegółowy	29
5.10	Dokumentacja pliku main.cpp	29
5.10.1	Opis szczegółowy	30
5.10.2	Dokumentacja funkcji	30
5.10.3	Dokumentacja zmiennych	30

5.11 Dokumentacja pliku Pliki.cpp	30
5.11.1 Opis szczegółowy	31
5.11.2 Dokumentacja funkcji	31
5.12 Dokumentacja pliku Pliki.hh	31
5.12.1 Opis szczegółowy	32
5.12.2 Dokumentacja funkcji	32
5.13 Dokumentacja pliku Statystyka.cpp	33
5.13.1 Opis szczegółowy	33
5.14 Dokumentacja pliku Statystyka.hh	33
5.14.1 Opis szczegółowy	33
5.15 Dokumentacja pliku Stoper.cpp	33
5.16 Dokumentacja pliku Stoper.hh	33
5.17 Dokumentacja pliku Wezel.hh	34
5.18 Dokumentacja pliku WezelAVL.hh	34
Indeks	35

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

IDrzew< typ >	15
DrzewoAVL< typ >	6
DrzewoAVLTest< typ >	9
DrzewoBinarne< typ >	10
DrzewoBinarneTest< typ >	13
IObserwator	17
Statystyka	20
IObserwowany	18
Benchmark< typ >	3
ITestable	19
DrzewoAVLTest< typ >	9
DrzewoBinarneTest< typ >	13
Stoper	22
Wezel< typ >	24
WezelAVL< typ >	26

2 Indeks klas

2.1 Lista klas

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

Benchmark < typ > Modeluje pojęcie Benchmarku	3
DrzewoAVL < typ > Definicja drzewa AVL	6
DrzewoAVLTest < typ > Testowalne Drzewo AVL	9
DrzewoBinarne < typ > DrzewoBinarne	10
DrzewoBinarneTest < typ > Drzewo binarne testowalne	13
IDrzew < typ > Definicja IDrzew	15
IObserwator Klasa IObserwator	17
IObserwowany Interfejs obserwowanego	18
ITestable Modeluje interfejs programu	19
Statystyka Modeluje pojęcie statystyki	20
Stoper Klasa Stoper	22
Wezel < typ > Klasa węzeł	24
WezelAVL < typ > Węzeł drzewa AVL	26

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Benchmark.hh Definicja klasy Benchmark	27
DrzewoAVL.hh	27
DrzewoAVLTest.hh	28
DrzewoBinarne.hh	28

DrzewoBinarneTest.hh	28
IDrzew.hh	28
IObserwator.hh	28
IObserwowany.hh	29
ITestable.hh	
Definicja klasy ITestable	29
main.cpp	
Moduł główny programu	29
Pliki.cpp	
Definicje funkcji obsługi plików	30
Pliki.hh	
Funkcje obsługi plików	31
Statystyka.cpp	
Zawiera definicję metod klasy Statystyka	33
Statystyka.hh	
Zawiera definicję klasy Statystyka	33
Stoper.cpp	33
Stoper.hh	33
Wezel.hh	34
WezelAVL.hh	34

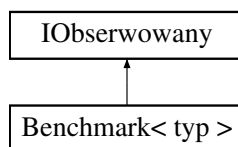
4 Dokumentacja klas

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 obiektu 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 obiekt klasy [Benchmark](#) i inicjuje nową statystykę dla obiektu

Parametry

in	<i>ileProb</i>	- ilość prób, które zostaną wykonane
in	<i>ileDanych</i>	- wskaźnik na tablice z licznosciami kolejnych serii
in	<i>ilePowtorzen</i>	- 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 obiektu

Parametry

in	<i>nowyObserwator</i>	- wskaźnik na obiekt będący obserwatorem
----	-----------------------	--

Implementuje [IObserwowany](#).

Definicja w linii 109 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 129 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 określonej liczbie serii i powtórzeniach pomiary zapisuje do pliku podanego przez użytkownika

Parametry

in	/	- obiekt klasy implementującej ITestable na której zostanie przeprowadzony test
in	<i>nazwaPliku</i>	- 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	<i>obserwator</i>	- wskaźnik na obserwatora który ma zostać usunięty
----	-------------------	--

Implementuje [IObserwowany](#).

Definicja w linii 120 pliku Benchmark.hh.

4.1.4 Dokumentacja atrybutów składowych

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

Tablica z licznosciami 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 serii

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 obiektu

Definicja w linii 57 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

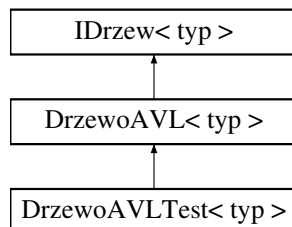
- [Benchmark.hh](#)

4.2 Dokumentacja szablonu klasy DrzewoAVL< typ >

Definicja drzewa AVL.

```
#include <DrzewoAVL.hh>
```

Diagram dziedziczenia dla DrzewoAVL< typ >



Metody publiczne

- `DrzewoAVL ()`
Konstruktor bezargumentowy.
- `~DrzewoAVL ()`
Destruktor.
- `void Insert (typ wartosc)`
Insert.
- `void Remove (typ wartosc)`
Remove.
- `WezelAVL< typ > * Search (typ wartosc)`
Search.
- `void CzyszcDrzewo ()`
CzyszcDrzewo.

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 Czyszc (WezelAVL< 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 pojęcie drzewa 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]
```

Konstruktor bezargumentowy zeruje liczbę węzłów i ustawia Korzeń 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 ( WezelAVL< typ > * wezel ) [inline],
[private]
```

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

Parametry

<i>wezel</i>	- 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, zwalnia po nich pamięć, następnie ustawia korzeń na NULL i Liczbę węzłów na 0.

Definicja w linii 455 pliku DrzewoAVL.hh.

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

Szuka węzła o najmniejszej wartości, począwszy od węzła podanego w argumencie.

Parametry

<i>in</i>	<i>poszukiwacz</i>	- węzeł startowy poszukiwań
-----------	--------------------	-----------------------------

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 argumencie wywołania.

Parametry

<i>in</i>	<i>poszukiwacz</i>	- węzeł którego poprzednik ma zostać znaleziony
-----------	--------------------	---

Zwracane wartości

<i>wskaźnik</i>	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

<i>in</i>	<i>wartosc</i>	- dana do dodania
-----------	----------------	-------------------

Implementuje [IDrzew< typ >](#).

Definicja w linii 265 pliku DrzewoAVL.hh.

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

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

Parametry

<i>in</i>	<i>A</i>	- wskaźnik do węzła głównego rotacji
-----------	----------	--------------------------------------

Definicja w linii 111 pliku DrzewoAVL.hh.

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

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

Parametry

<i>in</i>	<i>A</i>	- 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

<i>in</i>	<i>wartosc</i>	- przechowywana wartość do usunięcia z drzewa
-----------	----------------	---

Implementuje [IDrzew< typ >](#).

Definicja w linii 343 pliku DrzewoAVL.hh.

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

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

Parametry

<i>in</i>	<i>A</i>	- wskaźnik do węzła głównego rotacji
-----------	----------	--------------------------------------

Definicja w linii 143 pliku DrzewoAVL.hh.

4.2.3.10 `template<class typ> void DrzewoAVL< typ >::RR (WezełAVL< 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 437 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:

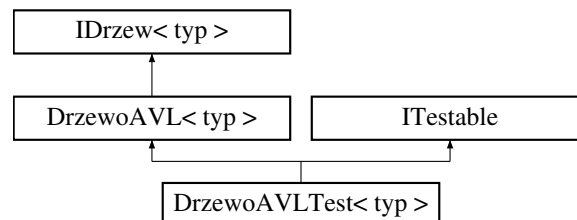
- [DrzewoAVL.hh](#)

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 umożliwiającymi jego testowanie.

The [DrzewoAVLTest](#) class

Klasa modelująca drzewo AVL z implemetacją metod niezbędnych 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	<i>k</i>	- ilość elementów dla których mają zostać wykonane obliczenia.
in	<i>nazwaPliku</i>	- 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	<i>nazwaPliku</i>	- nazwa pliku z danymi
in	<i>n</i>	- 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 obiekty wykorzystane do testów

Implementuje [ITestable](#).

Definicja w linii 75 pliku DrzewoAVLTest.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

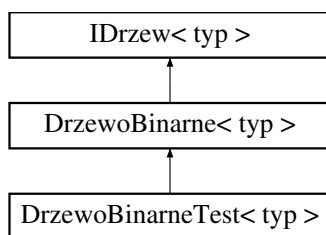
- [DrzewoAVLTest.hh](#)

4.4 Dokumentacja szablonu klasy DrzewoBinarne< typ >

[DrzewoBinarne](#).

```
#include <DrzewoBinarne.hh>
```

Diagram dziedziczenia dla DrzewoBinarne< typ >



Metody publiczne

- [DrzewoBinarne](#) ()
Konstruktor bezargumentowy.
- [~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 argumencie (z nim włącznie)

Parametry

<i>wezel</i>	- 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 najmniejszej wartości, poczynawszy od węzła podanego w arunemencie.

Parametry

<i>in</i>	<i>poszukiwacz</i>	- węzeł startowy poszukiwań
-----------	--------------------	-----------------------------

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 artumencie wywołania.

Parametry

<i>in</i>	<i>poszukiwacz</i>	- węzeł którego poprzednik ma zostać znaleziony
-----------	--------------------	---

Zwracane wartości

<i>wskaźnik</i>	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

<i>in</i>	<i>wartosc</i>	- 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

<i>in</i>	<i>wartosc</i>	- 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

<i>in</i>	<i>wartosc</i>	- 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:

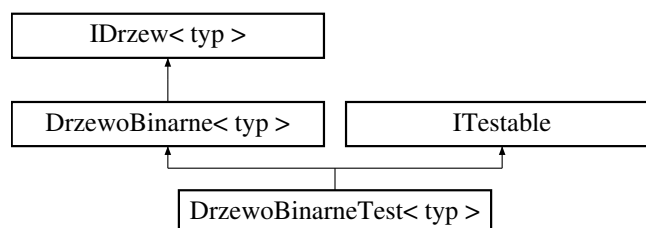
- [DrzewoBinarne.hh](#)

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 implementację testowalnego drzewa binarnego.

The DrzewoBinaneTest class

Drzewo binarne z zaimplementowanym interfejsem 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	<i>k</i>	- ilość elementów dla których mają zostać wykonane obliczenia.
in	<i>nazwaPliku</i>	- 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	<i>nazwaPliku</i>	- nazwa pliku z danymi
in	<i>n</i>	- 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 obiekty wykorzystane do testów

Implementuje [ITestable](#).

Definicja w linii 72 pliku DrzewoBinarneTest.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

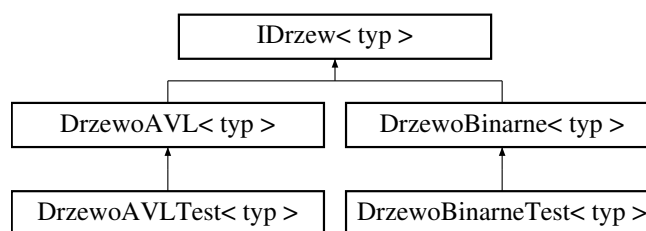
- [DrzewoBinarneTest.hh](#)

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 abstrakcyjna 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:

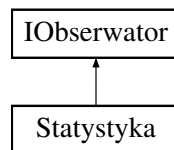
- [IDrzew.hh](#)

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 zawiera definicję klasy IObsereator.

The [IObserwator](#) class

Klasa modeluje interfejs obiektu 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 obiekcie obserwowanym.

Implementowany w [Statystyka](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

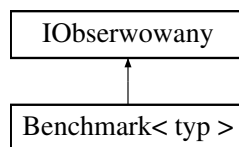
- [IObserwator.hh](#)

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 obiektu 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 obserwatorów danego obiektu.

Parametry

in	<i>nowyObserwator</i>	- wskaźnik na dodawanego obserwatora
----	-----------------------	--------------------------------------

Implementowany w [Benchmark< typ >](#).

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

Powiadamia obserwatorów 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 obiektu.

Parametry

in	obserwator	- obserwator do usunięcia z listy
----	------------	-----------------------------------

Implementowany w [Benchmark< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

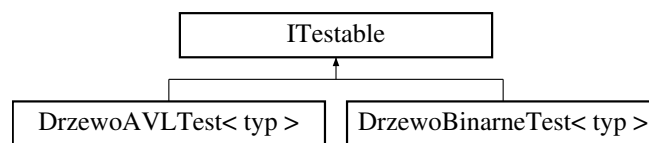
- [IObserwowany.hh](#)

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 wykonywanych 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	<i>nazwaPliku</i>	- nazwa pliku z danymi
in	<i>n</i>	- 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 obiekty wykorzystane do testów

Implementowany w [DrzewoAVLTest< typ >](#) i [DrzewoBinarneTest< typ >](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

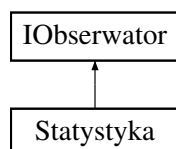
- [ITestable.hh](#)

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 parametrami.
- [~Statystyka](#) ()
Destruktor - zwalnia pamięć
- void [ZapiszStaty](#) (std::string nazwaPliku) const
Zapisuje statystykę 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 różnych 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 parametrami tworzy dynamiczne tablice przechowujące statystykę oraz wypełnia rozmiary prób.

Parametry

in	<i>iloscProb</i>	- liczbosc prob w ksperymentcie
in	<i>proby</i>	- tablica z licznosciami prób.

Definicja w linii 12 pliku Statystyka.cpp.

4.10.2.2 Statystyka::~Statystyka () [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablice 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 stoper odlicza to go zatrzymuje i sumuje czasy powtórzeń. Gdy nastąpi wykonanie wszystkich pomiarów w próbie to uzupełnia tablicę przechowywującą 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 statystykę 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	<i>nazwaPliku</i>	- nazwa pliku do którego ma zostać zapisana 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 pojedynczego testu.

Definicja w linii 65 pliku Statystyka.hh.

4.10.4.4 `unsigned int Statystyka::LicznikPowtorzen [private]`

Zlicza ilość wykonanych powtórzeń w danej próbie.

Definicja w linii 72 pliku Statystyka.hh.

4.10.4.5 `unsigned int Statystyka::LicznikProb [private]`

Zlicza ilość 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 pojedynczych 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 [CzasPoczątkowy](#)
- 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 destruktor

4.11.2.1 [Stoper::Stoper](#) ()

Konstruktor bezargumentowy 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 aktualnie liczy czy nie.

Zwracane wartości

<i>true</i>	- gdy odlicza
<i>false</i>	- 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::CzasPoczątkowy [private]

Czas w którym stoper zaczął odliczać.

Definicja w linii 27 pliku Stoper.hh.

4.11.4.3 bool Stoper::CzyLiczy [private]

Zmienna przechowuje wartość 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 struktury 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 destruktor

```
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 potomka 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](#) ()
- [~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 wykorzystywanego w drzewie AVL.

Węzeł drzewa AVL

Struktura nędać reprezentacją pojedynczego węzła w drzewie AVL.

Definicja w linii 17 pliku WezelAVL.hh.

4.13.2 Dokumentacja konstruktora i destruktor

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

Definicja w linii 49 pliku WezelAVL.hh.

```
4.13.2.2 template<class typ> WezelAVL< typ >::~~WezelAVL ( ) [inline]
```

Definicja w linii 56 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 potomka 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 "IObservowany.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 IObservowany.hh

```
#include "IObservator.hh"
```

Komponenty

- class [IObservowany](#)
Interfejs obserwowanego.

Definicje

- #define [IOBSERWOWANY_HH](#)

5.8.1 Dokumentacja definicji

5.8.1.1 #define IOBSERWOWANY_HH

Definicja w linii 2 pliku IObservowany.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`
Ilość 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 wykonuje serię 50 pomiarów czasu wykonania metody start dla różnych 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 38 pliku `main.cpp`.

5.10.3 Dokumentacja zmiennych

5.10.3.1 `const int ILOSC_POWTORZEN = 50`

Ilość powtórzeń danej próby

Definicja w linii 26 pliku `main.cpp`.

5.10.3.2 `const int ILOSC_PROB = 11`

Ilość prób = ilość rozmiarów prób

Definicja w linii 34 pliku `main.cpp`.

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

Definicja w linii 36 pliku `main.cpp`.

5.11 Dokumentacja pliku `Pliki.cpp`

Definicje funkcji obsługi plików.

```
#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 czyszcząc 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 związanych z obsługą plików.

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 podanego przez użytkownika następnie zapisuje wylosowane dane do pliku o nazwie "dane.dat"

Parametry

in	<i>n</i>	- ilość liczb do zapisania
in	<i>zakres</i>	- górny zakres wartości liczb

Definicja w linii 27 pliku Pliki.cpp.

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

Losuje *n* liczb z zakresu od 1 do podanego przez użytkownika następnie zapisuje wylosowane dane do pliku o nazwie "dane.dat"

Parametry

in	<i>n</i>	- ilość liczb do zapisania
in	<i>zakres</i>	- 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 się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyć
in	<i>plik</i>	- strumień powiązany 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 się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyć
in	<i>plik</i>	- strumień powiązany z plikiem

Definicja w linii 19 pliku Pliki.cpp.

5.12 Dokumentacja pliku Pliki.hh

Funkcje obsługi plików.

```
#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 czyszcząc 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 związanych z obsługą plików

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 podanego przez użytkownika następnie zapisuje wylosowane dane do pliku o nazwie "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 podanego przez użytkownika następnie zapisuje wylosowane dane do pliku o nazwie "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 się powiodło jeżeli nie to kończy program

Parametry

in	nazwaPliku	- nazwa pliku który chcemy otworzyć
in	plik	- strumień powiązany 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 się powiodło jeżeli nie to kończy program

Parametry

in	<i>nazwaPliku</i>	- nazwa pliku który chcemy otworzyc
in	<i>plik</i>	- 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 "IObservator.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

- ~DrzewoAVL
 - DrzewoAVL, [7](#)
- ~DrzewoBinarne
 - DrzewoBinarne, [12](#)
- ~Statystyka
 - Statystyka, [21](#)
- ~Wezel
 - Wezel, [25](#)
- ~WezelAVL
 - WezelAVL, [26](#)
- Aktualizuj
 - IObserwator, [17](#)
 - Statystyka, [21](#)
- Benchmark
 - Benchmark, [4](#)
 - DodajObserwatora, [4](#)
 - IleDanych, [5](#)
 - IlePowtorzen, [5](#)
 - IleProb, [5](#)
 - ListaObserwatorow, [5](#)
 - PowiadomObserwatorow, [4](#)
 - Test, [5](#)
 - UsunObserwatora, [5](#)
- Benchmark< typ >, [3](#)
- Benchmark.hh, [27](#)
- Czas
 - Statystyka, [21](#)
- CzasKoncowy
 - Stoper, [24](#)
- CzasPoczątkowy
 - Stoper, [24](#)
- CzyLiczy
 - Stoper, [24](#)
- CzyOdmierza
 - Stoper, [23](#)
- Czysc
 - DrzewoAVL, [7](#)
 - DrzewoBinarne, [12](#)
- CzyscDrzewo
 - DrzewoAVL, [7](#)
 - DrzewoBinarne, [12](#)
- DajPomiar
 - Stoper, [23](#)
- Dana
 - Wezel, [25](#)
 - WezelAVL, [26](#)
- DodajObserwatora
 - Benchmark, [4](#)
 - IObserwowany, [18](#)
- DrzewoAVL
 - ~DrzewoAVL, [7](#)
 - Czysc, [7](#)
 - CzyscDrzewo, [7](#)
 - DrzewoAVL, [7](#)
 - DrzewoAVL, [7](#)
 - FindMin, [7](#)
 - FindSuccessor, [7](#)
 - Insert, [8](#)
 - Korzen, [9](#)
 - LL, [8](#)
 - LR, [8](#)
 - LiczbaWezlow, [9](#)
 - RL, [8](#)
 - RR, [8](#)
 - Remove, [8](#)
 - Search, [9](#)
- DrzewoAVL< typ >, [6](#)
- DrzewoAVL.hh, [27](#)
- DrzewoAVLTest
 - Start, [10](#)
 - WczytajDane, [10](#)
 - Zwolnij, [10](#)
- DrzewoAVLTest< typ >, [9](#)
- DrzewoAVLTest.hh, [28](#)
- DrzewoBinarne
 - ~DrzewoBinarne, [12](#)
 - Czysc, [12](#)
 - CzyscDrzewo, [12](#)
 - DrzewoBinarne, [12](#)
 - DrzewoBinarne, [12](#)
 - FindMin, [12](#)
 - FindSuccessor, [12](#)
 - Insert, [13](#)
 - Korzen, [13](#)
 - LiczbaWezlow, [13](#)
 - Remove, [13](#)
 - Search, [13](#)
- DrzewoBinarne< typ >, [10](#)
- DrzewoBinarne.hh, [28](#)
- DrzewoBinarneTest
 - Start, [14](#)
 - WczytajDane, [14](#)
 - Zwolnij, [15](#)
- DrzewoBinarneTest< typ >, [13](#)
- DrzewoBinarneTest.hh, [28](#)
- FindMin
 - DrzewoAVL, [7](#)
 - DrzewoBinarne, [12](#)
- FindSuccessor
 - DrzewoAVL, [7](#)
 - DrzewoBinarne, [12](#)
- IDrzew
 - Insert, [15](#)
 - Remove, [17](#)
- IDrzew< typ >, [15](#)
- IDrzew.hh, [28](#)

ILOSC_POWTORZEN
 main.cpp, 30
 ILOSC_PROB
 main.cpp, 30
 IOBSERWOWANY_HH
 IObserwowany.hh, 29
 IObserwator, 17
 Aktualizuj, 17
 IObserwator.hh, 28
 IObserwowany, 18
 DodajObszerwatora, 18
 PowiadomObszerwatorow, 18
 UsunObszerwatora, 18
 IObserwowany.hh, 29
 IOBSERWOWANY_HH, 29
 ITestable, 19
 Start, 19
 WczytajDane, 19
 Zwolnij, 20
 ITestable.hh, 29
 IleDanych
 Benchmark, 5
 IlePowtorzen
 Benchmark, 5
 IleProb
 Benchmark, 5
 Statystyka, 22
 IloscPowtorzen
 Statystyka, 22
 Insert
 DrzewoAVL, 8
 DrzewoBinarne, 13
 IDrzew, 15

 Korzen
 DrzewoAVL, 9
 DrzewoBinarne, 13

 LL
 DrzewoAVL, 8
 LR
 DrzewoAVL, 8
 Lewy
 Wezel, 25
 WezelAVL, 26
 LiczbaWezlów
 DrzewoAVL, 9
 DrzewoBinarne, 13
 LicznikPowtorzen
 Statystyka, 22
 LicznikProb
 Statystyka, 22
 ListaObszerwatorow
 Benchmark, 5
 LosujIntRandDoPliku
 Pliki.cpp, 31
 Pliki.hh, 32
 LosujIntRoznacoDoPliku
 Pliki.cpp, 31
 Pliki.hh, 32

 Pliki.hh, 32

 main
 main.cpp, 30
 main.cpp, 29
 ILOSC_POWTORZEN, 30
 ILOSC_PROB, 30
 main, 30
 MojStoper
 Statystyka, 22

 OtworzPlikIn
 Pliki.cpp, 31
 Pliki.hh, 32
 OtworzPlikOut
 Pliki.cpp, 31
 Pliki.hh, 32

 Pliki.cpp, 30
 LosujIntRandDoPliku, 31
 LosujIntRoznacoDoPliku, 31
 OtworzPlikIn, 31
 OtworzPlikOut, 31
 Pliki.hh, 31
 LosujIntRandDoPliku, 32
 LosujIntRoznacoDoPliku, 32
 OtworzPlikIn, 32
 OtworzPlikOut, 32
 PowiadomObszerwatorow
 Benchmark, 4
 IObserwowany, 18
 Prawy
 Wezel, 25
 WezelAVL, 26
 Proba
 Statystyka, 22

 RL
 DrzewoAVL, 8
 RR
 DrzewoAVL, 8
 Remove
 DrzewoAVL, 8
 DrzewoBinarne, 13
 IDrzew, 17
 Reset
 Stoper, 24
 Rodzic
 Wezel, 25
 WezelAVL, 27

 Search
 DrzewoAVL, 9
 DrzewoBinarne, 13
 Start
 DrzewoAVLTest, 10
 DrzewoBinarneTest, 14
 ITestable, 19
 Stoper, 24

Statystyka, [20](#)
 ~Statystyka, [21](#)
 Aktualizuj, [21](#)
 Czas, [21](#)
 IleProb, [22](#)
 IloscPowtorzen, [22](#)
 LicznikPowtorzen, [22](#)
 LicznikProb, [22](#)
 MojStoper, [22](#)
 Proba, [22](#)
 Statystyka, [21](#)
 SumaCzasuProby, [22](#)
 ZapiszStaty, [21](#)
Statystyka.cpp, [33](#)
Statystyka.hh, [33](#)
Stop
 Stoper, [24](#)
Stoper, [22](#)
 CzasKoncowy, [24](#)
 CzasPoczatkowy, [24](#)
 CzyLiczy, [24](#)
 CzyOdmierza, [23](#)
 DajPomiar, [23](#)
 Reset, [24](#)
 Start, [24](#)
 Stop, [24](#)
 Stoper, [23](#)
Stoper.cpp, [33](#)
Stoper.hh, [33](#)
SumaCzasuProby
 Statystyka, [22](#)

Test
 Benchmark, [5](#)

UsunObserwatora
 Benchmark, [5](#)
 IObservowany, [18](#)

WczytajDane
 DrzewoAVLTest, [10](#)
 DrzewoBinarneTest, [14](#)
 ITestable, [19](#)
Wezel
 ~Wezel, [25](#)
 Dana, [25](#)
 Lewy, [25](#)
 Prawy, [25](#)
 Rodzic, [25](#)
 Wezel, [25](#)
Wezel< typ >, [24](#)
Wezel.hh, [34](#)
WezelAVL
 ~WezelAVL, [26](#)
 Dana, [26](#)
 Lewy, [26](#)
 Prawy, [26](#)
 Rodzic, [27](#)
 WezelAVL, [26](#)

WezelAVL, [26](#)
 WspRownowagi, [27](#)
WezelAVL< typ >, [26](#)
WezelAVL.hh, [34](#)
WspRownowagi
 WezelAVL, [27](#)

ZapiszStaty
 Statystyka, [21](#)
Zwolnij
 DrzewoAVLTest, [10](#)
 DrzewoBinarneTest, [15](#)
 ITestable, [20](#)