PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Pt, 29 maj 2015 14:06:21

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:

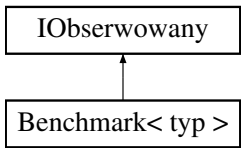
4 Dokumentacja klas

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, std::string const nazwaPlikuDane[])

  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 112 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 132 pliku Benchmark.hh.

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

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

Parametry

in	*I*	- objekt klasy na której zostanie przeprowadzony test
in	*nazwaPlikuDane*	- 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 123 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 struktury Lista< typ >::Element

Modeluje jeden element Listy.

Metody publiczne

- Element (typ k)

  Konstruktor daną przekazywaną w argumencie.

Atrybuty publiczne

- typ wartosc

  Wartosc Elementu.
- Element * nastepny

  Wskaźnik na kolejny Element Listy.

4.2.1 Opis szczegółowy

template<class typ>struct Lista< typ >::Element

Modeluje jeden nierozłączny element listy - przechowywaną daną oraz wskaźnik na następny element;

Definicja w linii 34 pliku Lista.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 template<class typ> Lista< typ >::Element::Element (typ *k*) [inline]

Konstruktor zapisujący w Elemecie na końcu Listy daną podaną w argumencie i ustawiający wkaźnik na NULL

Parametry

in	*k*	- dana która ma zostać dodana na koniec Listy

Definicja w linii 60 pliku Lista.hh.

4.2.3 Dokumentacja atrybutów składowych

4.2.3.1 template<class typ> Element* Lista< typ >::Element::nastepny

Wskaźnik na kolejny Element Listy

Definicja w linii 49 pliku Lista.hh.

4.2.3.2 template<class typ> typ Lista< typ >::Element::wartosc

Wartość Elementu - przechowywanej wartości przez dany Element listy

Definicja w linii 42 pliku Lista.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

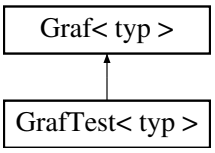
- Lista.hh

4.3 Dokumentacja szablonu klasy Graf< typ >

Graf.

`#include <Graf.hh>`

Diagram dziedziczenia dla Graf< typ >



Metody publiczne

- Wierzcholek< typ > * DajWierzcholek (const typ wartosc) const
  DajWierzcholek.
- void DodajWierzcholek (const typ wartosc)
  DodajWierzcholek.
- void DodajKrawedz (const typ v, const typ w)
  DodajKrawedz.
- void UsunWierzcholek (typ wartosc)
  UsunWierzcholek.
- void ResetLabels ()
  ResetLabels.
- void SciezkaDFS (Lista< typ > *sciezka, Wierzcholek< typ > *poczatkowyW, const typ koncowyW)
  SciezkaDFS.
- void SciezkaBFS (Lista< typ > *sciezka, Wierzcholek< typ > *poczatkowyW, const typ koncowyW)
  SciezkaBFS.

Metody prywatne

- Wierzcholek< typ > * PrzeciwnyW (Wierzcholek< typ > *w, Krawedz< typ > *k)
  PrzeciwnyW.
- void SetLabelW (Wierzcholek< typ > *w, status stat)
  SetLabelW.
- void SetLabelE (Krawedz< typ > *e, status stat)
  SetLabel.
- status GetLabelW (Wierzcholek< typ > *w)
  GetLabelW.
- status GetLabelE (Krawedz< typ > *e)
  GetLabelE.

Atrybuty prywatne

- Lista< Wierzcholek< typ > * > ListaWierzcholkow
  ListaWierzcholkow.
- Lista< Krawedz< typ > * > ListaKrawedzi
  ListaKrawedzi.
- Lista< Lista< Wierzcholek< typ > * > > ListySasiedztwa
  ListySasiedztwa.
- bool znaleziono = false
  znaleziono

4.3.1 Opis szczegółowy

template<class typ>class Graf< typ >

Plik zawira definicję klasy graf

The Graf class

Klasa modeluje poęcie Grafu.

Definicja w linii 19 pliku Graf.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 template<class typ > Wierzcholek<typ>* Graf< typ >::DajWierzcholek (const typ *wartosc*) const [inline]

Szuka wierzchołka przechowującego daną wartość i zwraca wskaźnik na niego.

Parametry

in	*wartosc*	- wartośc przechowywana przez szukany wierzcholek

Zwracane wartości

-	wskaźnik na wierzchołek lub null w przypakdu nie znalezienia

Definicja w linii 133 pliku Graf.hh.

4.3.2.2 template<class typ > void Graf< typ >::DodajKrawedz (const typ *v*, const typ *w*) [inline]

Dodaje krrawędź łączącą dwa wierzchołki do grafu.

Parametry

in	*v*	- wskaźnik na jeden z łączonych wierzchołków
in	*w*	- wskaźnik na drugi wierzchołek

Definicja w linii 166 pliku Graf.hh.

4.3.2.3 template<class typ > void Graf< typ >::DodajWierzcholek (const typ *wartosc*) [inline]

Dodaje wierzchołek przechowujący daną wartośc do grafu. Jeżli wierzchołek o danej wartości już istnieje to jest pomijany.

Parametry

in	*wartosc*	- wartośc jaka przechowuje nowy wierzchołek

Definicja w linii 152 pliku Graf.hh.

4.3.2.4 template<class typ > status Graf< typ >::GetLabelE (Krawedz< typ > * *e*) [inline], [private]

Zwraca aktualną flagę gałęzi

Parametry

in	*w*	- wskaźnik na gałąź, której flaga jest czytana

Zwracane wartości

-	aktualna flaga gałęzi

Definicja w linii 116 pliku Graf.hh.

4.3.2.5 template<class typ > status Graf< typ >::GetLabelW (Wierzcholek< typ > * w) [inline], [private]

Zwraca aktualną flagę wierzchołka

Parametry

in	*w*	- wskaźnik na wierzchołek, którego flaga jest czytana

Zwracane wartości

-	aktualna flaga wierzchołka

Definicja w linii 104 pliku Graf.hh.

4.3.2.6 template<class typ > Wierzcholek<typ>* Graf< typ >::PrzeciwnyW (Wierzcholek< typ > * w, Krawedz< typ > * k) [inline], [private]

Metoda zwraca przeciwny Wierzcholek do wierzchołka w względem krawędzi k.

Parametry

in	*w*	- wskaźnik na wierzchołek do którego szukany jest przeciwny
in	*k*	- wskaźnik na krawędź która łączy wierzchołki

Zwracane wartości

-	zwraca wskaźnik na przeciwległy wierzchołek

Definicja w linii 64 pliku Graf.hh.

4.3.2.7 template<class typ > void Graf< typ >::ResetLabels ( ) [inline]

Resetuje flagi wszystkich elementów grafu ustawiając jako nieodwiedzony.

Definicja w linii 210 pliku Graf.hh.

4.3.2.8 template<class typ > void Graf< typ >::SciezkaBFS (Lista< typ > * sciezka, Wierzcholek< typ > * poczatkowyW, const typ koncowyW) [inline]

Wyznacza ścierzkę z jednego wierzchołka do drugiego wykorzystując algorytm BFS.

Parametry

in	*sciezka*	- wskaźnik na listę w której będą przechowywane kolejna wierzchołki ścieżki
in	*poczatkowyW*	- wskaźnik na początkowy wierzchołek ścieżki
in	*koncowyW*	- wartość przechowywana przez ostatni wierzchołek ścieżki

Definicja w linii 268 pliku Graf.hh.

4.3.2.9 template<class typ > void Graf< typ >::SciezkaDFS (Lista< typ > * sciezka, Wierzcholek< typ > * poczatkowyW, const typ koncowyW) [inline]

Wyznacza ścierzkę z jednego wierzchołka do drugiego wykorzystując algorytm DFS.

Parametry

in	*sciezka*	- wskaźnik na listę w której będą przechowywane kolejna wierzchołki ścieżki
in	*poczatkowyW*	- wskaźnik na początkowy wierzchołek ścieżki
in	*koncowyW*	- wartość przechowywana przez ostatni wierzchołek ścieżki

Definicja w linii 228 pliku Graf.hh.

4.3.2.10 template<class typ > void Graf< typ >::SetLabelE (Krawedz< typ > * *e,* status *stat*) [inline], [private]

Ustawia flagę krawedzi zgodnie z podanym statusem {nieodwiedzony, odwiedzony, powrotny, poprzeczny}

Parametry

in	*w*	- wskaźnik na krawedz
in	*stat*	- status do ustawienia

Definicja w linii 92 pliku Graf.hh.

4.3.2.11 template<class typ > void Graf< typ >::SetLabelW (Wierzcholek< typ > * *w,* status *stat*) [inline], [private]

Ustawia flagę wierzchołka zgodnie z podanym statusem {nieodwiedzony, odwiedzony, powrotny, poprzeczny}

Parametry

in	*w*	- wskaźnik na wierzchołek
in	*stat*	- status do ustawienia

Definicja w linii 79 pliku Graf.hh.

4.3.2.12 template<class typ > void Graf< typ >::UsunWierzcholek (typ *wartosc*) [inline]

Usuwa dany wierzcholek z grafu wraz z przyległymi do niego krawędziamy

Parametry

in	*wartosc*	- wartość przechowywana przez wierzchołek do usunięcia

Definicja w linii 188 pliku Graf.hh.

4.3.3 Dokumentacja atrybutów składowych

4.3.3.1 template<class typ > Lista< Krawedz<typ>* > Graf< typ >::ListaKrawedzi [private]

Pole przechowuje listę wsaźników na krawędzi grafu.

Definicja w linii 34 pliku Graf.hh.

4.3.3.2 template<class typ > Lista< Wierzcholek<typ>* > Graf< typ >::ListaWierzcholkow [private]

Pole przechowuje listę wskaźników na wierzchołki grafu.

Definicja w linii 26 pliku Graf.hh.

4.3.3.3 template<class typ > Lista< Lista< Wierzcholek<typ>* > > Graf< typ >::ListySasiedztwa [private]

Pole przechowujące listy sąsiedztwa wierzchołków grafu.

Definicja w linii 42 pliku Graf.hh.

4.3.3.4 template<class typ > bool Graf< typ >::znaleziono = false [private]

Zmienna mocnicza przy rekurencyjnym algorytmie DFS szukania ścieżki

Definicja w linii 49 pliku Graf.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

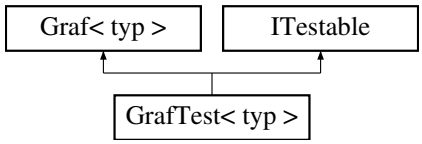
- Graf.hh

4.4 Dokumentacja szablonu klasy GrafTest< typ >

GrafTest.

`#include <GrafTest.hh>`

Diagram dziedziczenia dla GrafTest< typ >



Metody publiczne

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

  Wczytanie danych z pliku.
- void Start (const unsigned int w1, const unsigned int w2)

  Wykonanie części obliczeniowej programu.
- void Reset ()

  Reset.
- void Zwolnij (std::string const nazwaPliku, const unsigned int n)

  Zwalnia pamięć po teście.

4.4.1 Opis szczegółowy

template<class typ>class GrafTest< typ >

Plik zawiera definicję klasy GrafTest

GrafTest

Klasa modelująca pojecie grafu z zaimplementowanymi metodami niezbędnymi do przeprowadzneia testów.

Definicja w linii 20 pliku GrafTest.hh.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 template<class typ> void GrafTest< typ >::Reset ( ) [inline], [virtual]

Resetuje flagi.

Implementuje ITestable.

Definicja w linii 85 pliku GrafTest.hh.

4.4.2.2 template<class typ> void GrafTest< typ >::Start (const unsigned int *w1*, const unsigned int *w2*) [inline], [virtual]

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

Parametry

in	*w1*	- wartość początkowego wierzchołka
in	*w2*	- wartość końcowego wierzchołka

Implementuje ITestable.

Definicja w linii 71 pliku GrafTest.hh.

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

Wczytuje zadaną ilość danych do przetworzenia z pliku o zadanej nazwie. Plik musi wyglądać następująco:

- pierwsza dana to pierwszy wierzchołek,
- n-1 trójek "nowy_wierzcholek wierzcholek_krawedzi wierzcholek_krawedzi"
- reszta pliku to pary istniejących wierzchołków tworzących krawędzie

Parametry

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

Implementuje ITestable.

Definicja w linii 38 pliku GrafTest.hh.

4.4.2.4 template<class typ> void GrafTest< typ >::Zwolnij (std::string const *nazwaPliku,* const unsigned int *n*) [inline], [virtual]

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

param[in] nazwaPliku - plik z danymi param[in] n - ilosc danych

Implementuje ITestable.

Definicja w linii 98 pliku GrafTest.hh.

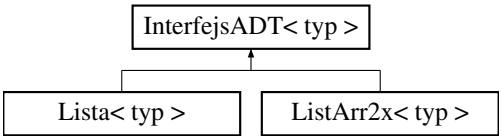
Dokumentacja dla tej klasy została wygenerowana z pliku:

- GrafTest.hh

4.5 Dokumentacja szablonu klasy InterfejsADT< typ >

`#include <InterfejsADT.hh>`

Diagram dziedziczenia dla InterfejsADT< typ >



Metody publiczne

- virtual void push (const typ dana, const unsigned int pole)=0

  Dodaje kolejny element.
- virtual void pop (const unsigned int pole)=0

  Pobiera element.

- virtual unsigned int size () const =0

  Liczność elemetów.

4.5.1 Opis szczegółowy

template<class typ>class InterfejsADT< typ >

\ brief Definiuje interfejs użytkownika

Definiuje interfejs użytkownika dla listy, stosu i kolejki.

Definicja w linii 12 pliku InterfejsADT.hh.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 template<class typ> virtual void InterfejsADT< typ >::pop (const unsigned int *pole*) [pure virtual]

Pobiera element z typu danych

Parametry

in	*pole*	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola z ktore pobiera element

Zwracane wartości

zwraca	wartość danego elementu

Implementowany w Lista< typ >, Lista< Lista< Wierzcholek< typ > * > >, Lista< Krawedz< typ > * >, Lista< Wierzcholek< typ > * > i ListArr2x< typ >.

4.5.2.2 template<class typ> virtual void InterfejsADT< typ >::push (const typ *dana,* const unsigned int *pole*) [pure virtual]

Dodaje kolejny element do typu danych

Parametry

in	*dana*	- element który chcemy dorzucić do naszego typu
in	*pole*	- !!!DOSTEPNE TYLKO DLA LISTY!!! nr pola na które chcemy dodać element

Implementowany w Lista< typ >, Lista< Lista< Wierzcholek< typ > * > >, Lista< Krawedz< typ > * >, Lista< Wierzcholek< typ > * > i ListArr2x< typ >.

4.5.2.3 template<class typ> virtual unsigned int InterfejsADT< typ >::size ( ) const [pure virtual]

Informuje o liczności elementów obecnie przechowywanych

Zwracane wartości

zwraca	ilość przechowywanych elementów

Implementowany w Lista< typ >, Lista< Lista< Wierzcholek< typ > * > >, Lista< Krawedz< typ > * >, Lista< Wierzcholek< typ > * > i ListArr2x< typ >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

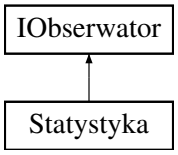
- InterfejsADT.hh

4.6 Dokumentacja klasy IObserwator

Klasa IObserwator.

`#include <IObserwator.hh>`

Diagram dziedziczenia dla IObserwator



Metody publiczne

- virtual void Aktualizuj ()=0

  Aktualizuj.

4.6.1 Opis szczegółowy

Plik zawira definicję klasy IObsereator.

The IObserwator class

Klasa modeluje interfejs objektu będącego obserwatorem.

Definicja w linii 15 pliku IObserwator.hh.

4.6.2 Dokumentacja funkcji składowych

4.6.2.1 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:

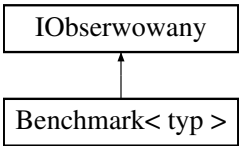
- IObserwator.hh

4.7 Dokumentacja klasy IObserwowany

The IObserwowany class.

`#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.

4.7.1 Opis szczegółowy

Klasa czysto wirtualna modelująca interfejs objektu obserwowanego.

Definicja w linii 17 pliku IObserwowany.hh.

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 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 >.

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

Powiadamia obseratorów o wydarzeniu.

Implementowany w Benchmark< typ >.

4.7.2.3 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:

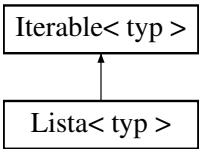
- IObserwowany.hh

4.8 Dokumentacja szablonu klasy Iterable< typ >

Definicja Iterable.

```
#include <Iterable.hh>
```

Diagram dziedziczenia dla Iterable< typ >



Metody publiczne

- virtual typ operator[] (const size_t pole) const =0

  operator []

4.8.1 Opis szczegółowy

template<class typ>class Iterable< typ >

Plik zawiera definicje interfejsu Iterable

The Iterable class

Klasa modeluje interfejs umożliwiajacy przeglądanie kontenera oraz uzystkiwanie referencji do jego ostatniego pola co jest wymagane w obecnej implementacji tablicy asocjacyjnej

Definicja w linii 21 pliku Iterable.hh.

4.8.2 Dokumentacja funkcji składowych

4.8.2.1 template<class typ> virtual typ Iterable< typ >::operator[] (const size_t *pole*) const [pure virtual]

Przeciążenie operatora [] w celu umożliwienia przeglądania kontenera

Parametry

in	*pole*	- indeks elementu

Zwracane wartości

-	zwraca wartośc znajdującą się na danym indeksie

Implementowany w Lista< typ >, Lista< Lista< Wierzcholek< typ > * > >, Lista< Krawedz< typ > * > i Lista< Wierzcholek< typ > * >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

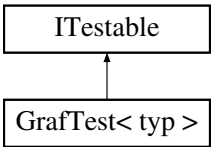
- Iterable.hh

4.9 Dokumentacja klasy ITestable

Modeluje interfejs tetsowy programu.

`#include <ITestable.hh>`

Diagram dziedziczenia dla ITestable



Metody publiczne

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

  Wczytanie danych z pliku.
- virtual void Start (const unsigned int w1, const unsigned int w2)=0

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

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

  Zwalnia pamięć po teście.

4.9.1 Opis szczegółowy

Modeluje interfejs testowy do programów wykonywanch w ramach kursu.

Definicja w linii 25 pliku ITestable.hh.

4.9.2 Dokumentacja funkcji składowych

4.9.2.1 virtual void ITestable::Reset ( ) [pure virtual]

Resetuje flagi.

Implementowany w GrafTest< typ >.

4.9.2.2 virtual void ITestable::Start (const unsigned int *w1*, const unsigned int *w2*) [pure virtual]

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

Parametry

in	*w1*	- pierwsza wartość
in	*w2*	- druga wartość

Implementowany w GrafTest< typ >.

4.9.2.3 virtual void ITestable::WczytajDane (std::string const *nazwaPliku*, const 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 GrafTest< typ >.

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

Zwalnia pamięć zajmowaną przez objekty wykorzytsane do testów

param[in] nazwaPliku - plik z danymi param[in] n - ilosc danych

Implementowany w GrafTest< typ >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ITestable.hh

4.10 Dokumentacja szablonu struktury Krawedz< typ >

Krawedz.

`#include <Krawedz.hh>`

Metody publiczne

- Krawedz ()
  Krawedz.

Atrybuty publiczne

- Wierzcholek< typ > * WierzcholekLewy

WierzcholekLewy.

- Wierzcholek< typ > * WierzcholekPrawy

  WierzcholekPrawy.

- status Label

  Label.

4.10.1 Opis szczegółowy

template<class typ>struct Krawedz< typ >

Definicja wierzcholka.

Plik zawiera definicję struktury Krawedz

Krawedz

Struktura modeluje pojęcie krawędzi wierzchołka.

Plik zawiera definicję struktury Wierzchołek

Definicja w linii 18 pliku Krawedz.hh.

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 template<class typ> Krawedz< typ >::Krawedz ( ) [inline]

Konstruktor ustwiający wskaźniki na wierzchołki na wartość NULL.

Definicja w linii 46 pliku Krawedz.hh.

4.10.3 Dokumentacja atrybutów składowych

4.10.3.1 template<class typ> status Krawedz< typ >::Label

Flaga krawędzi niezbędna dla algorytmów przechodzenia przez graf.

Definicja w linii 39 pliku Krawedz.hh.

4.10.3.2 template<class typ> Wierzcholek<typ>* Krawedz< typ >::WierzcholekLewy

Wskaźnik na jeden z wierzchołków przyległych do krawędzi.

Definicja w linii 25 pliku Krawedz.hh.

4.10.3.3 template<class typ> Wierzcholek<typ>* Krawedz< typ >::WierzcholekPrawy

Wskaźnik na drugi z wierzchołków przyległych do krawędzi.

Definicja w linii 32 pliku Krawedz.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

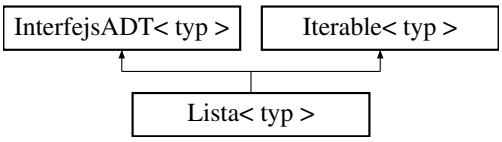
- Krawedz.hh

4.11 Dokumentacja szablonu klasy Lista< typ >

Modeluje pojęcie listy.

`#include <Lista.hh>`

Diagram dziedziczenia dla Lista< typ >



Komponenty

- struct Element

  Modeluje jeden element Listy.

Metody publiczne

- Lista ()

  Konstruktor puste listy.
- void Zwolnij ()

  Destruktor listy.
- void push (const typ dana, const unsigned int pole)

  Dodaje daną do Listy.
- void pop (const unsigned int pole)

  Usuwa element z Listy.
- unsigned int size () const

  Sprawdza rozmiar Listy.
- typ operator[] (const size_t pole) const

  Wyciąga wartość elementu Listy.
- void Remove (const typ dana)

  Remove.

Atrybuty prywatne

- Element ∗ Poczatek

  Wskaźnik na pierwszy element Listy.
- Element ∗ Koniec

  Wzkaźnik na ostatni element listy.
- unsigned int Rozmiar

  Aktualny rozmiar Listy.

4.11.1 Opis szczegółowy

template<class typ>class Lista< typ >

Modeluje pojęcie listy zadeklarowanego w szablonie typu Uwaga! Listę indeksujemy od 0.

Definicja w linii 25 pliku Lista.hh.

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 template<class typ> Lista< typ >::Lista ( ) [inline]

Konstruktor bezargumentowy pustej listy tworzy objekt z wskaźnikiem początek pokazującym na NULL.

Definicja w linii 99 pliku Lista.hh.

4.11.3 Dokumentacja funkcji składowych

4.11.3.1 template<class typ> typ Lista< typ >::operator[] (const size_t *pole*) const [inline], [virtual]

Wyłuskuje wartość danego elementu z Listy

Parametry

in	*pole*	- "indeks" z którego chcemy pobrać wartość indeksujemy od 0!

Zwracane wartości

-	zwraca wartość elementu z danego pola lub '-1' w przypadku błedu

Implementuje Iterable< typ >.

Definicja w linii 248 pliku Lista.hh.

4.11.3.2 template<class typ> void Lista< typ >::pop (const unsigned int *pole*) [inline], [virtual]

Usuwa interesujący nas element z Listy. Jeżeli chcesz usunąć pierwszy element wywołaj pole nr '0'. Dla ostatniego elementu wywołaj pole nr 'Lista.size()-1'.

Parametry

in	*pole*	- numer elementu Listy z którego chcemy pobrać daną

Zwracane wartości

zwraca	wartość danego elementu listy lub '-1' w przypadku błędu

Implementuje InterfejsADT< typ >.

Definicja w linii 191 pliku Lista.hh.

4.11.3.3 template<class typ> void Lista< typ >::push (const typ *dana*, const unsigned int *pole*) [inline], [virtual]

Dodaje daną podaną jako pierwszy argument wywołania na określone drugim argumentem miejsce w Liście

Parametry

in	*dana*	- dana którą chcemy dodać do listy
in	*pole*	- numer elementu listy na który chcemy dodać daną (sieze() jeżeli na koniec)

Implementuje InterfejsADT< typ >.

Definicja w linii 143 pliku Lista.hh.

4.11.3.4 template<class typ> void Lista< typ >::Remove (const typ *dana*) [inline]

Usuwa z listy element przechowujący daną wartość

Parametry

in	*dana*	- wartość przechowywana prze zelement do usunięcia

Definicja w linii 265 pliku Lista.hh.

4.11.3.5 template<class typ> unsigned int Lista< typ >::size ( ) const [inline], [virtual]

Sprawdza ile aktualnie elementów znajduję się na Liście

Zwracane wartości

zwraca	ilosć elementów znadjących się aktualnie na liście

Implementuje InterfejsADT< typ >.

Definicja w linii 234 pliku Lista.hh.

4.11.3.6 template<class typ> void Lista< typ >::Zwolnij ( ) [inline]

Zwalnia zaalokowana przez liste pamiec

Zwalnia pamięć

Zwalnia pamięć zajmowaną przez listę

Definicja w linii 123 pliku Lista.hh.

4.11.4 Dokumentacja atrybutów składowych

4.11.4.1 template<class typ> Element* Lista< typ >::Koniec [private]

Wskaźnik na ostatni element listy

Definicja w linii 80 pliku Lista.hh.

4.11.4.2 template<class typ> Element* Lista< typ >::Poczatek [private]

Wskaźnik na pierwszy element Listy

Definicja w linii 72 pliku Lista.hh.

4.11.4.3 template<class typ> unsigned int Lista< typ >::Rozmiar [private]

Przechowuje aktualną ilość Elemenetów znajujących się na Liście

Definicja w linii 87 pliku Lista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

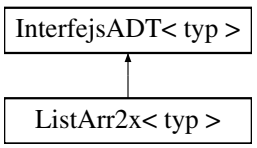
- Lista.hh

4.12 Dokumentacja szablonu klasy ListArr2x< typ >

Modeluje pojęcie Listy (array)

`#include <ListArr2x.hh>`

Diagram dziedziczenia dla ListArr2x< typ >



Metody publiczne

- ListArr2x ()

  Konstruktor bezarumentowy.
- void push (const typ dana, const unsigned int pole)

  Dodaje element do ListyArr2x.

- void pop (const unsigned int pole)

  Pobiera element z ListyArr2x.
- unsigned int size () const

  Wielkość listy.
- void Zwolnij ()

  Zwalnia pamięć
- typ operator[] (const size_t pole) const

  operator []
- void Remove (const typ dana)

  Remove.

Metody prywatne

- void UsunZListy (const unsigned int pole)

  UsunZListy.
- void DodajDoListy (const typ dana, const unsigned int pole)

  DodajDoListy.

Atrybuty prywatne

- typ * tab

  Wkaźnik na dynamiczną tablicę
- unsigned int RozmiarT

  Rozmiar tablicy.
- unsigned int RozmiarL

  Rozmiar Listy.

4.12.1 Opis szczegółowy

template<class typ>class ListArr2x< typ >

Modeluje pojęcie Listy opartej na dynamicznej tablicy. Dodając elementy zwiększa tablicę dwukrotnie, jeżeli brakuje miejsca. a

Definicja w linii 18 pliku ListArr2x.hh.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 template<class typ > ListArr2x< typ >::ListArr2x ( ) [inline]

Kontruktor alokujący tablicę jednoelementową z której będzie tworzona lista

Definicja w linii 86 pliku ListArr2x.hh.

4.12.3 Dokumentacja funkcji składowych

4.12.3.1 template<class typ > void ListArr2x< typ >::DodajDoListy (const typ *dana*, const unsigned int *pole*) [inline], [private]

Dodaje daną do listy na określony indeks

Parametry

dana	- wartość która ma zostać umieszczona na liście
pole	- indeks pola na którym ma zostać umieszczona wartość

Definicja w linii 66 pliku ListArr2x.hh.

4.12.3.2 template<class typ > typ ListArr2x< typ >::operator[] (const size_t *pole*) const [inline]

Przeciążenie operatora [] w celu umożliwienia przeglądania listy

Parametry

in	*i*	- indeks elementu

Zwracane wartości

-	zwraca wartośc znajdującą się na danym indeksie

Definicja w linii 191 pliku ListArr2x.hh.

4.12.3.3 template<class typ > void ListArr2x< typ >::pop (const unsigned int *pole*) [inline], [virtual]

Pobiera element z ListyArr2x usuwając go z niej i zmniejszając rozmiar o połowę w przypadku przekroczenia stosunku 1:4 (RozmiarL:RozmiarT)

param[in] - pole - nr pola z którgo chcemy pobrać element (indeksowane od 0)

Implementuje InterfejsADT< typ >.

Definicja w linii 137 pliku ListArr2x.hh.

4.12.3.4 template<class typ > void ListArr2x< typ >::push (const typ *dana,* const unsigned int *pole*) [inline], [virtual]

Dodaje nowy element do ListyArr2x

Parametry

in	*dana*	- element który chcemy umieścić na liście
in	*pole*	- nr pola na którym chcemy umieścić element jeżeli chcesz umieścić na początku listy podaj wartość 0, na końcu warość size()

Implementuje InterfejsADT< typ >.

Definicja w linii 102 pliku ListArr2x.hh.

4.12.3.5 template<class typ > void ListArr2x< typ >::Remove (const typ *dana*) [inline]

Usuwa z listy element przechowujący daną wartość

Parametry

in	*dana*	- wartość przechowywana prze zelement do usunięcia

Definicja w linii 202 pliku ListArr2x.hh.

4.12.3.6 template<class typ > unsigned int ListArr2x< typ >::size ( ) const [inline], [virtual]

Informuje o ilości elementów znajdujących się na LiścieArr1

Zwracane wartości

-	zwraca liczbę elementów ListyArr1

Implementuje InterfejsADT< typ >.

Definicja w linii 170 pliku ListArr2x.hh.

4.12.3.7 template<class typ > void ListArr2x< typ >::UsunZListy (const unsigned int *pole*) [inline], [private]

Usuwa z listy element o podanym indeksie

Parametry

in	*pole*	- indeks elementu do usunięcia.

Definicja w linii 48 pliku ListArr2x.hh.

4.12.3.8 template<class typ > void ListArr2x< typ >::Zwolnij ( ) [inline]

Zwalnia pamięć zaalokowaną przez ListArr2x

Definicja w linii 178 pliku ListArr2x.hh.

4.12.4 Dokumentacja atrybutów składowych

4.12.4.1 template<class typ > unsigned int ListArr2x< typ >::RozmiarL [private]

Aktualny rozmiar ListyArr2x

Definicja w linii 39 pliku ListArr2x.hh.

4.12.4.2 template<class typ > unsigned int ListArr2x< typ >::RozmiarT [private]

Aktualny rozmiar tablicy.

Definicja w linii 32 pliku ListArr2x.hh.

4.12.4.3 template<class typ > typ* ListArr2x< typ >::tab [private]

Wskaźnik na dynamiczną tablicę tworzącą ListęArr2x

Definicja w linii 25 pliku ListArr2x.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

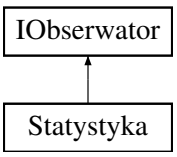
- ListArr2x.hh

4.13 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.

4.13.1 Opis szczegółowy

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

Definicja w linii 27 pliku Statystyka.hh.

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 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.
in	*ilePowtorzen*	- ilość powtórzeń każdego rozmiaru próby

Definicja w linii 12 pliku Statystyka.cpp.

4.13.2.2 Statystyka::∼Statystyka ( ) [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablicy przechowujące statystykę.

Definicja w linii 109 pliku Statystyka.hh.

4.13.3 Dokumentacja funkcji składowych

4.13.3.1 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.

4.13.3.2 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.

4.13.4 Dokumentacja atrybutów składowych

4.13.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 51 pliku Statystyka.hh.

4.13.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 35 pliku Statystyka.hh.

4.13.4.3 unsigned int Statystyka::IloscPowtorzen [private]

Przechowuje ilość wykonywanych powtórzeń pojedyńczego testu.

Definicja w linii 65 pliku Statystyka.hh.

4.13.4.4 unsigned int Statystyka::LicznikPowtorzen [private]

Zlicza ilosć wykonanych powtórzeń w danej próbie.

Definicja w linii 72 pliku Statystyka.hh.

4.13.4.5 unsigned int Statystyka::LicznikProb [private]

Zlicza ilosć prób wykonanych prób.

Definicja w linii 79 pliku Statystyka.hh.

4.13.4.6 Stoper* Statystyka::MojStoper [private]

Stoper wykorzystywany do pomiaru czasu.

Definicja w linii 86 pliku Statystyka.hh.

4.13.4.7 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 43 pliku Statystyka.hh.

4.13.4.8 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

4.14 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.

4.14.1 Opis szczegółowy

Plik zawiera definicję klasy Stoper.

The Stoper class

Klasa modeluje stoper niezbędny do odliczania czasu.

Definicja w linii 18 pliku Stoper.hh.

4.14.2 Dokumentacja konstruktora i destruktora

4.14.2.1 Stoper::Stoper ( )

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

Definicja w linii 3 pliku Stoper.cpp.

4.14.3 Dokumentacja funkcji składowych

4.14.3.1 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.

4.14.3.2 double Stoper::DajPomiar ( ) const

Wyłuskuje czas pomiaru w ms.

Zwracane wartości

zwrca	czas pomiaru wyrażon w ms

Definicja w linii 25 pliku Stoper.cpp.

4.14.3.3 void Stoper::Reset ( )

Resetuje stoper.

Definicja w linii 19 pliku Stoper.cpp.

4.14.3.4 void Stoper::Start ( )

Uruchamia odliczanie czasu.

Definicja w linii 9 pliku Stoper.cpp.

4.14.3.5 void Stoper::Stop ( )

Zatrzymuje odliczanie czasu.

Definicja w linii 14 pliku Stoper.cpp.

4.14.4 Dokumentacja atrybutów składowych

4.14.4.1 double Stoper::CzasKoncowy [private]

Czas w którym odliczanie czasu zostało zatrzymane.

Definicja w linii 32 pliku Stoper.hh.

4.14.4.2 double Stoper::CzasPoczatkowy [private]

Czas w którym stoper zaczął odliczać.

Definicja w linii 25 pliku Stoper.hh.

4.14.4.3 bool Stoper::CzyLiczy [private]

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

Definicja w linii 40 pliku Stoper.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Stoper.hh
- Stoper.cpp

4.15 Dokumentacja szablonu struktury Wierzcholek< typ >

Wierzcholek.

`#include <Wierzcholek.hh>`

Metody publiczne

- ~Wierzcholek ()

  Destruktor.

Atrybuty publiczne

- typ Dana

  Dana.
- Lista< Krawedz< typ > * > ListaKrawedziV

  ListaKrawedziV.
- status Label

  Label.

4.15.1 Opis szczegółowy

template<class typ>struct Wierzcholek< typ >

Struktura modeluje pojęcie wierzchołka grafu.

Definicja w linii 23 pliku Wierzcholek.hh.

4.15.2 Dokumentacja konstruktora i destruktora

4.15.2.1 template<class typ> Wierzcholek< typ >::~Wierzcholek ( ) [inline]

Destrunktor wierzchołka usuwający krawędzie przyległe.

Definicja w linii 51 pliku Wierzcholek.hh.

4.15.3 Dokumentacja atrybutów składowych

4.15.3.1 template<class typ> typ Wierzcholek< typ >::Dana

Wartość przechowywana przez wierzchołek.

Definicja w linii 30 pliku Wierzcholek.hh.

4.15.3.2 template<class typ> status Wierzcholek< typ >::Label

Flaga wierzchołka niezbędna dla algorytmów przechodzenia przez graf.

Definicja w linii 44 pliku Wierzcholek.hh.

4.15.3.3 template<class typ> Lista< Krawedz<typ>* > Wierzcholek< typ >::ListaKrawedziV

Lista wskaźników na krawędzi odchodzące od wierzchołka.

Definicja w linii 37 pliku Wierzcholek.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Wierzcholek.hh

5 Dokumentacja plików

5.1 Dokumentacja pliku Benchmark.hh

Definicja klasy Benchmark.

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

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 Graf.hh

```
#include "Krawedz.hh"
#include "ListArr2x.hh"
```

Komponenty

- class Graf< typ >
  Graf.

5.3 Dokumentacja pliku GrafTest.hh

```
#include "Graf.hh"
#include "Iterable.hh"
```

Komponenty

- class GrafTest< typ >

  GrafTest.

5.4 Dokumentacja pliku InterfejsADT.hh

Komponenty

- class InterfejsADT< typ >

5.5 Dokumentacja pliku IObserwator.hh

Komponenty

- class IObserwator

  Klasa IObserwator.

5.6 Dokumentacja pliku IObserwowany.hh

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

Komponenty

- class IObserwowany

  The IObserwowany class.

Definicje

- #define IOBSERWOWANY_HH

  Interfejs obserwowanego.

5.6.1 Dokumentacja definicji

5.6.1.1 #define IOBSERWOWANY_HH

W pliku zawarta jest definicja interfejsu obserwowanego

Definicja w linii 8 pliku IObserwowany.hh.

5.7 Dokumentacja pliku Iterable.hh

```
#include <iostream>
```

Komponenty

- class Iterable< typ >

  Definicja Iterable.

5.8 Dokumentacja pliku ITestable.hh

Definicja klasy ITestable.

```
#include <iostream>
#include "Pliki.hh"
```

Komponenty

- class ITestable

  Modeluje interfejs tetsowy programu.

5.8.1 Opis szczegółowy

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

Definicja w pliku ITestable.hh.

5.9 Dokumentacja pliku Krawedz.hh

```
#include "Wierzcholek.hh"
```

Komponenty

- struct Krawedz< typ >

  Krawedz.

5.10 Dokumentacja pliku Lista.hh

Eefinicja klasy Lista.

```
#include "InterfejsADT.hh"
#include "Pliki.hh"
#include "Iterable.hh"
```

Komponenty

- class Lista< typ >

  Modeluje pojęcie listy.
- struct Lista< typ >::Element

  Modeluje jeden element Listy.

5.10.1 Opis szczegółowy

Plik zawiera definicję klasy lista ujętej w szablon typu przchowywanych zmiennych więc zawiera też definicję metod klasy.

Definicja w pliku Lista.hh.

5.11 Dokumentacja pliku ListArr2x.hh

Definicja klasy ListArr1.

```
#include "InterfejsADT.hh"
```

Komponenty

- class ListArr2x< typ >

  Modeluje pojęcie Listy (array)

5.11.1 Opis szczegółowy

Plik zawiera definicję klasy ListaArr2x ujętej w szablon typu wraz z jej składowymi metofdami.

Definicja w pliku ListArr2x.hh.

5.12 Dokumentacja pliku main.cpp

Moduł główny programu.

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

Definicje

- #define ILOSC_POWTORZEN 10

  Ilośc powtórzeń danej próby.
- #define ILOSC_PROB 9

  Ilość prób.

Funkcje

- int main (int argc, char ∗argv[])

5.12.1 Opis szczegółowy

Program wkonuje serię 10 pomiarów czasu wykonania metody start (badanie czasu zapisu i odczytu do/z Tablicy Asocjacyjnej dla różncyh wielkości problemu obliczeniowego. Jako plik wynikowy otrzymujemy plik z czasami poświęconymi przez program na zapis/odczyt n danych z tablicy.

WYMAGANIA: Plik z danymi musi być w formacie takim, że każda linia to kolejno "klucz wartość"

Klucze muszą być sześcioznakowymi ciągami stringów składających się wyłącznie z małych liter.

Wartości mogą być dowolnym intem

OBSŁUGA PROGRAMU: Aby wywołać program należy w lini poleceń wywołać jego nazę np: "./a.out"

Definicja w pliku main.cpp.

5.12.2 Dokumentacja definicji

5.12.2.1 #define ILOSC_POWTORZEN 10

Ilośc powtórzeń danej próby

Definicja w linii 39 pliku main.cpp.

5.12.2.2 #define ILOSC_PROB 9

Ilość prób = ilość rozmiarów prób

Definicja w linii 47 pliku main.cpp.

5.12.3 Dokumentacja funkcji

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

Definicja w linii 49 pliku main.cpp.

5.13 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 LosujIntDoPliku (const unsigned int n, const unsigned int zakres)

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

  LosujGrafIntDoPliku.

5.13.1 Opis szczegółowy

Plik zawiera definicje funkcji zwiazanych z obsluga plikow.

Definicja w pliku Pliki.cpp.

5.13.2 Dokumentacja funkcji

5.13.2.1 void LosujGrafIntDoPliku (const unsigned int *w,* const unsigned int *e,* const unsigned int *zakres*)

Generuje plik zawierający losowo wygenerowane, niepowtarzające się wierzchołki i krawędzie(mogące się powtarzać) tworzące spójny graf. Wierzchołki są reprezentowane przez liczby typu całkowitoliczbowego.

Schemat pliku:

"pierwszy_wierzchołek nowy_wierzchołek nowy_wierzchołekTK wierzcholekTK"

gdzie trójki (nie licząc pierwszego_wierzchołka) powtarzają się tyle razy ile ma być wierzchołków pomniejszonych o 1. , gdzie zaimek TK oznacza wierzchołek z grupy wcześniej wygenerowanych wierzchołków do którego zaczepiony

zostaje koniec krawędzi między (nowy_wierzcholekTK i wierzcholekTK). Reszte pliku stanowią pary wierzchołków tworzące nowe krawędzie.

Ilość krawędzi musi być większa lub równa ilości wierzchołków.

Parametry

w	- zadana ilość wierzchołków grafu
e	- zadana ilość grawędzi
zakres	- zakres liczb wartości wierzchołków

Definicja w linii 72 pliku Pliki.cpp.

5.13.2.2 void LosujIntDoPliku (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.

5.13.2.3 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.

5.13.2.4 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.

5.14 Dokumentacja pliku Pliki.hh

Funkcje obslugi plikow.

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include "Lista.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 LosujIntDoPliku (const unsigned int n, const unsigned int zakres)

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

  LosujGrafIntDoPliku.

5.14.1 Opis szczegółowy

Plik zawiera deklaracje funkcji zwiazanych z obsuga plikow

Definicja w pliku Pliki.hh.

5.14.2 Dokumentacja funkcji

5.14.2.1 void LosujGrafIntDoPliku (const unsigned int *w,* const unsigned int *e,* const unsigned int *zakres*)

Generuje plik zawierający losowo wygenerowane, niepowtarzające się wierzchołki i krawędzie(mogące się powtarzać) tworzące spójny graf. Wierzchołki są reprezentowane przez liczby typu całkowitoliczbowego.

Schemat pliku:

"pierwszy_wierzchołek nowy_wierzchołek nowy_wierzchołekTK wierzcholekTK"

gdzie trójki (nie licząc pierwszego_wierzchołka) powtarzają się tyle razy ile ma być wierzchołków pomniejszonych o 1. , gdzie zaimek TK oznacza wierzchołek z grupy wcześniej wygenerowanych wierzchołków do którego zaczepiony zostaje koniec krawędzi między (nowy_wierzcholekTK i wierzcholekTK). Reszte pliku stanowią pary wierzchołków tworzące nowe krawędzie.

Ilość krawędzi musi być większa lub równa ilości wierzchołków.

Parametry

w	- zadana ilość wierzchołków grafu
e	- zadana ilość grawędzi
zakres	- zakres liczb wartości wierzchołków

Definicja w linii 72 pliku Pliki.cpp.

5.14.2.2 void LosujIntDoPliku (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.

5.14.2.3 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.

5.14.2.4 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.

5.15 Dokumentacja pliku status.hh

Wyliczenia

- enum status { nieodwiedzony, odwiedzony, powrotny, poprzeczny }

  The status enum.

5.15.1 Dokumentacja typów wyliczanych

5.15.1.1 enum status

Definicja typu wyliczeniowego status, który jest wykorzystywany przez algorytmy przejćia grafu jako flagi ustawiane na wierzchołkach i krawędziach.

Wartości wyliczeń

nieodwiedzony

odwiedzony

powrotny

poprzeczny

Definicja w linii 11 pliku status.hh.

5.16 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy Statystyka.

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

5.16.1 Opis szczegółowy

Plik zawiera definicję metod klasy Statystyka.

Definicja w pliku Statystyka.cpp.

5.17 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.

5.17.1 Opis szczegółowy

Zawiera definicję klasy Statystyka

Definicja w pliku Statystyka.hh.

5.18 Dokumentacja pliku Stoper.cpp

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

5.19 Dokumentacja pliku Stoper.hh

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

Komponenty

- class Stoper

  Klasa Stoper.

5.20 Dokumentacja pliku Wierzcholek.hh

```
#include "Lista.hh"
#include "status.hh"
```

Komponenty

- struct Krawedz< typ >

  Krawedz.
- struct Wierzcholek< typ >

  Wierzcholek.

Skorowidz