PAMSI_LAB

Wygenerowano przez Doxygen 1.8.6

Śr, 18 mar 2015 21:36:24

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 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.

4.1.1 Opis szczegółowy

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

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

Definicja w linii 36 pliku Lista.cpp.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 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 62 pliku Lista.cpp.

4.1.3 Dokumentacja atrybutów składowych

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

Wskaźnik na kolejny Element Listy

Definicja w linii 51 pliku Lista.cpp.

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

Wartość Elementu - przechowywanej wartości przez dany Element listy

Definicja w linii 44 pliku Lista.cpp.

Dokumentacja dla tej struktury została wygenerowana z pliku:

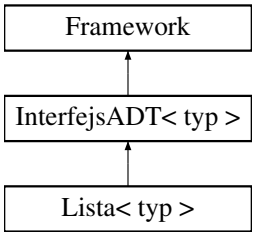
- Lista.cpp

4.2 Dokumentacja klasy Framework

Modeluje interfejs programu.

`#include <Framework.hh>`

Diagram dziedziczenia dla Framework



Metody publiczne

- virtual void WczytajDane (const char *nazwaPliku, unsigned int n)=0

  Wczytanie danych z pliku.

- virtual void Start (const unsigned int k)=0

  Wykonanie części obliczeniowej programu.

4.2.1 Opis szczegółowy

Modeluje interfejs do programów wykonywanch w ramach kursu.

Definicja w linii 24 pliku Framework.hh.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 virtual void Framework::Start (const unsigned int *k*) [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 Lista< typ > i InterfejsADT< typ >.

4.2.2.2 virtual void Framework::WczytajDane (const char * *nazwaPliku,* unsigned int *n*) [pure virtual]

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

Parametry

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

Implementowany w Lista< typ > i InterfejsADT< typ >.

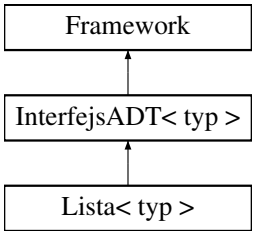
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Framework.hh

4.3 Dokumentacja szablonu klasy InterfejsADT< typ >

`#include <InterfejsADT.hh>`

Diagram dziedziczenia dla InterfejsADT< typ >



Metody publiczne

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

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

  Pobiera element.

- virtual unsigned int size ()=0

  Liczność elemetów.
- void WczytajDane (const char *nazwaPliku, unsigned int n)=0

  Wczytanie danych z pliku.
- void Start (const unsigned int k)=0

  Wykonanie części obliczeniowej programu.

4.3.1 Opis szczegółowy

template<class typ>class InterfejsADT< typ >

\ brief Definiuje interfejs użytkownika

Definiuje interfejs użytkownika dla listy, stosu i kolejki.

Definicja w linii 13 pliku InterfejsADT.hh.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 template<class typ> virtual InterfejsADT< typ >::~InterfejsADT ( ) [pure virtual]

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 template<class typ> virtual void InterfejsADT< typ >::pop (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 >.

4.3.3.2 template<class typ> virtual void InterfejsADT< typ >::push (typ *dana,* 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 >.

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

Informuje o liczności elementów obecnie przechowywanych

Zwracane wartości

zwraca	ilość przechowywanych elementów

Implementowany w Lista< typ >.

4.3.3.4 template<class typ> void InterfejsADT< typ >::Start (const unsigned int *k*) [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.

Implementuje Framework.

Implementowany w Lista< typ >.

4.3.3.5 template<class typ> void InterfejsADT< typ >::WczytajDane (const char * *nazwaPliku*, unsigned int *n*) [pure virtual]

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

Parametry

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

Implementuje Framework.

Implementowany w Lista< typ >.

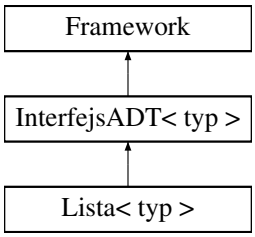
Dokumentacja dla tej klasy została wygenerowana z pliku:

- InterfejsADT.hh

4.4 Dokumentacja szablonu klasy Lista< typ >

Modeluje pojęcie listy.

Diagram dziedziczenia dla Lista< typ >



Komponenty

- struct Element

  Modeluje jeden element Listy.

Metody publiczne

- Lista ()

  Konstruktor puste listy.
- ~Lista ()

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

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

  Usuwa element z Listy.
- unsigned int size ()

  Sprawdza rozmiar Listy.

- void WczytajDane (const char *nazwaPliku, unsigned int n)

  Wczytuje dane z pliku.
- void Start (const unsigned int k)

  Proces obliczeniowy.

Atrybuty prywatne

- Element * Poczatek

  Wskaźnik na pierwszy element Listy.
- Element * Koniec

  Wzkaźnik na ostatni element listy.
- unsigned int Rozmiar

  Aktualny rozmiar Listy.

4.4.1 Opis szczegółowy

template<class typ>class Lista< typ >

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

Definicja w linii 27 pliku Lista.cpp.

4.4.2 Dokumentacja konstruktora i destruktora

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

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

Definicja w linii 101 pliku Lista.cpp.

4.4.2.2 template<class typ > Lista< typ >::~Lista ( )

Zwalnia zaalokowana przez liste pamiec

Definicja w linii 252 pliku Lista.cpp.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 template<class typ> void Lista< typ >::pop (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

Implementuje InterfejsADT< typ >.

Definicja w linii 174 pliku Lista.cpp.

4.4.3.2 template<class typ> void Lista< typ >::push (typ *dana*, 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

Implementuje InterfejsADT< typ >.

Definicja w linii 122 pliku Lista.cpp.

4.4.3.3 template<class typ> unsigned int Lista< typ >::size ( ) [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 218 pliku Lista.cpp.

4.4.3.4 template<class typ> void Lista< typ >::Start (const unsigned int *k*) [inline], [virtual]

Wykonuje proces oblcizeniowy, którego czas wykonania jest mierzony na potrzeby laboratoriów PAMSI W tym wypakdu tworzy Listę k elementową wypełnioną stałą liczbą '3'.

Parametry

in	*k*	- ilość danych dla których ma zostać przeprowadzona precedura obnliczeniowa

Implementuje InterfejsADT< typ >.

Definicja w linii 244 pliku Lista.cpp.

4.4.3.5 template<class typ> void Lista< typ >::WczytajDane (const char * *nazwaPliku,* unsigned int *n*) [inline], [virtual]

Wczytuje dane zamieszczone w pliku do Listy. Każdą nową daną umieszcza na końcu listy.

Parametry

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

Implementuje InterfejsADT< typ >.

Definicja w linii 230 pliku Lista.cpp.

4.4.4 Dokumentacja atrybutów składowych

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

Wskaźnik na ostatni element listy

Definicja w linii 82 pliku Lista.cpp.

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

Wskaźnik na pierwszy element Listy

Definicja w linii 74 pliku Lista.cpp.

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

Przechowuje aktualną ilość Elemenetów znajujących się na Liście

Definicja w linii 89 pliku Lista.cpp.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Lista.cpp

4.5 Dokumentacja klasy Statystyka

Modeluje pojęcie statystyki.

`#include <Statystyka.hh>`

Metody publiczne

- Statystyka (const unsigned int iloscProb, unsigned int ∗proby)
  Konstruktor z dwoma pramametrami.
- ∼Statystyka ()
  Destruktor - zwaknia pamięć
- double & operator[] (unsigned int i)
  Indeksuje tablicę czasową
- void ZapiszStaty ()
  Zapisuje statysykę do pliku.

Atrybuty prywatne

- unsigned int IleProb
  Ilość prób.
- unsigned int ∗ Proba
  Tablica z rozmiarami prób.
- double ∗ Czas
  Średni czas wykonania danej próby.

4.5.1 Opis szczegółowy

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

Definicja w linii 22 pliku Statystyka.hh.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 Statystyka::Statystyka (const unsigned int *iloscProb,* unsigned int ∗ *proby*)

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

Parametry

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

Definicja w linii 13 pliku Statystyka.cpp.

4.5.2.2 Statystyka::∼Statystyka ( ) [inline]

Zwalnia pamięć zaalokowaną na dynamiczne tablicy przechowujące statystykę.

Definicja w linii 68 pliku Statystyka.hh.

4.5.3 Dokumentacja funkcji składowych

4.5.3.1 double& Statystyka::operator[] (unsigned int *i*) [inline]

Zwraca referencję do i-tego indeksu tablicy czasowej.

Parametry

in	*i*	- indeks tablicy czasowej

Zwracane wartości

Czas[i]	referencja do wybranego indeksu

Definicja w linii 80 pliku Statystyka.hh.

4.5.3.2 void Statystyka::ZapiszStaty ( )

Zapisuje statystystykę do pliku o nazwie "statystyka.dat". Pierwsza linia pliku to wielkości prób druga to średnie czasy wykonania podane w ms;

Definicja w linii 21 pliku Statystyka.cpp.

4.5.4 Dokumentacja atrybutów składowych

4.5.4.1 double* Statystyka::Czas [private]

wskaźnik na tablica ze średnimi czasami wykonania kolejnych prób.

Definicja w linii 46 pliku Statystyka.hh.

4.5.4.2 unsigned int Statystyka::IleProb [private]

Ilość prób do utworzenia statystyki

Definicja w linii 30 pliku Statystyka.hh.

4.5.4.3 unsigned int* Statystyka::Proba [private]

Wskaźnik na tablicę zawierającą wielkości danych prób.

Definicja w linii 38 pliku Statystyka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Statystyka.hh
- Statystyka.cpp

5 Dokumentacja plików

5.1 Dokumentacja pliku Framework.hh

Definicja klasy Framework.

```
#include <iostream>
```

Komponenty

- class Framework

  Modeluje interfejs programu.

5.1.1 Opis szczegółowy

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

Definicja w pliku Framework.hh.

5.2 Dokumentacja pliku InterfejsADT.hh

```
#include "Framework.hh"
```

Komponenty

- class InterfejsADT< typ >

5.3 Dokumentacja pliku Lista.cpp

Eefinicja klasy Lista.

```
#include "../inc/InterfejsADT.hh"
#include "../inc/Pliki.hh"
#include <ctime>
#include <iostream>
#include <cstdlib>
```

Komponenty

- class Lista< typ >
  Modeluje pojęcie listy.
- struct Lista< typ >::Element
  Modeluje jeden element Listy.

Definicje

- #define LISTA_HH

Funkcje

- int main ()

5.3.1 Opis szczegółowy

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

Definicja w pliku Lista.cpp.

5.3.2 Dokumentacja definicji

5.3.2.1 #define LISTA_HH

Definicja w linii 2 pliku Lista.cpp.

5.3.3 Dokumentacja funkcji

5.3.3.1 int main ( )

Definicja w linii 262 pliku Lista.cpp.

5.4 Dokumentacja pliku main.cpp

Moduł główny programu.

```
#include "../src/Lista.cpp"
#include "../inc/Statystyka.hh"
#include <ctime>
#include "../inc/InterfejsADT.hh"
#include "../inc/Framework.hh"
```

Definicje

- #define ILOSC_POWTORZEN 10
- #define ILOSC_PROB 5

Funkcje

- int main (int argc, char ∗argv[])

5.4.1 Opis szczegółowy

Program wkonuje serię 10 pomiarów czasu wykonania metody start dla różncyh wielkości problemu obliczeniowego. Dane do obliczeń wczytuje z pliku o nazwie podanej w pierwszym argumencie wywołania programu, a statystykę pomiarów zapisuje do pliku o nazwie "statystyka.dat".

OBSŁUGA PROGRAMU: Aby wywołać program należy w lini poleceń wywołać jego nazę i jako pierwszy argument podać nazwę pliku z miliardem danych w formacie int np: "./a.out dane.dat" Jeżeli nie posiadamy takiego pliku to podczas wywoływania programu należe podać jako pierwszy argument nazwę pliku "dane.dat" i dodatkowo jakikolwiek drugui argument, spowoduje to utworzenie pliku z danymi o nazwie "dane.dat" przed częścią obliczeniową programu. Przykład wywoałania z tworzeniem pliku z danymi: "./a.out dane.dat l"

Definicja w pliku main.cpp.

5.4.2 Dokumentacja definicji

5.4.2.1 #define ILOSC_POWTORZEN 10

Definicja w linii 30 pliku main.cpp.

5.4.2.2 #define ILOSC_PROB 5

Definicja w linii 31 pliku main.cpp.

5.4.3 Dokumentacja funkcji

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

Definicja w linii 33 pliku main.cpp.

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

  Zapisuje n losowych liczb(int) do pliku.

5.5.1 Opis szczegółowy

Plik zawiera definicje funkcji zwiazanych z obsluga plikow.

Definicja w pliku Pliki.cpp.

5.5.2 Dokumentacja funkcji

5.5.2.1 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 19 pliku Pliki.cpp.

5.5.2.2 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.

5.6 Dokumentacja pliku Pliki.hh

Funkcje obslugi plikow.

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

Funkcje

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

  Otwiera plik do odczytu.

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

5.6.1 Opis szczegółowy

Plik zawiera deklaracje funkcji zwiazanych z obsuga plikow

Definicja w pliku Pliki.hh.

5.6.2 Dokumentacja funkcji

5.6.2.1 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 19 pliku Pliki.cpp.

5.6.2.2 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.

5.7 Dokumentacja pliku Statystyka.cpp

Zawiera definicję metod klasy Statystyka.

```
#include "../inc/Statystyka.hh"
#include <fstream>
#include <cstdlib>
```

5.7.1 Opis szczegółowy

Plik zawiera definicję metod klasy Statystyka.

Definicja w pliku Statystyka.cpp.

5.8 Dokumentacja pliku Statystyka.hh

Zawiera definicję klasy Statystyka.

```
#include <iostream>
```

Komponenty

- class Statystyka

Modeluje pojęcie statystyki.

5.8.1 Opis szczegółowy

Zawiera definicję klasy Statystyka

Definicja w pliku Statystyka.hh.

Skorowidz