LAB4

0.1

Wygenerowano przez Doxygen 1.8.6

Śr, 29 kwi 2015 23:01:01

Spis treści

Rozdział 1

Lab1

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja szablonu klasy Benchmark< T >

`#include <Benchmark.hh>`

Metody publiczne

- **Benchmark** ()
- **Benchmark** (int roz, unsigned long int max, T *Struktura=nullptr)
- ~**Benchmark** ()
- void **UstalRozmiarTablicyZlozonosciObliczeniowej** (int roz)
- std::ostream & **GenerujLiczbyZmiennoprzecinkowe** (long int **rozmiar**, std::ostream &strum)
- std::ostream & **GenerujLiczbyCalkowiteLosowe** (long int **rozmiar**, std::ostream &strum)
- void **TransformacjaBoxa_Mullera** (float *a)
- void **StworzLiczbyOdniesienia** (int p[], int ile)
- void **Testuj** (int MaxIloscDanych, void(*wsk_fun)(T *, int))
- std::ostream & **ZapiszWynikiZlozonosciObliczeniowej** (std::ostream &Strm)

Atrybuty publiczne

- unsigned long int ** **ZlozonoscObliczeniowa**
- T * **LiczbyGaussowe**
- unsigned long int **IloscDanych**
- int **rozmiar**

4.1.1 Opis szczegółowy

template<typename T>class Benchmark< T >

Definicja w linii 16 pliku Benchmark.hh.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 template<typename T > Benchmark< T >::Benchmark ( )

brief Konstruktor bezparametryczny

Konstruktor bezparametryczny klasy **Benchmark** (str. 7) Ustawia wszystkie zmienne i wskazniki na wartosc 0

Definicja w linii 97 pliku Benchmark.hh.

4.1.2.2 template<typename T > Benchmark< T >::Benchmark (int *roz,* unsigned long int *max,* T * *Struktura =* nullptr)

brief Konstruktor z 2 argumentami int, unsigned long int

Konstruktor parametryczny klasy **Benchmark** (str. 7)

Parametry

in	*roz*	- typ int, rozmiar tablicy ZlozonoscObliczeniowa,
in	*max-*	unsigned long int, ilosc liczb zmiennoprzecinkowych tablicy LiczbyGaussowe

Definicja w linii 82 pliku Benchmark.hh.

4.1.2.3 template<typename T > Benchmark< T >::~Benchmark ( )

brief destruktor klasy **Benchmark** (str. 7)

Definicja w linii 106 pliku Benchmark.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 template<typename T > std::ostream & Benchmark< T >::GenerujLiczbyCalkowiteLosowe (long int *rozmiar,* std::ostream & *strum*)

brief Funkcja generuje liczby typu double z rozkladu Gaussa

Funkcja generuje liczby typu int o rozkladzie Gaussa i zapisuje do strumienia

Parametry

in	*rozmiar*	- long int, ilosc wygenerowanych elementow
in	*&strum*	- referencja do strumienia wyjsciowego

Zwraca

Zwraca referencje do strumienia wyjsciowego

Warunek wstępny

Prawidlowe dzialanie funkcji TransformacjaBoxa_Mullera

Definicja w linii 204 pliku Benchmark.hh.

4.1.3.2 template<typename T > std::ostream & Benchmark< T >::GenerujLiczbyZmiennoprzecinkowe (long int *rozmiar,* std::ostream & *strum*)

brief Funkcja generuje liczby typu double z rozkladu Gaussa

Funkcja generuje liczby typu float o rozkladzie Gaussa i zapisuje do strumienia

Parametry

in	*rozmiar*	- long int, ilosc wygenerowanych elementow
in	*&strum*	- referencja do strumienia wyjsciowego

Zwraca

Zwraca referencje do strumienia wyjsciowego

Warunek wstępny

Prawidlowe dzialanie funkcji TransformacjaBoxa_Mullera

Definicja w linii 175 pliku Benchmark.hh.

4.1.3.3 template<typename T> void Benchmark< T >::StworzLiczbyOdniesienia (int *p[],* int *ile*)

brief Funkcja tworzy wygodne liczby odniesienia dla danej funkcji testowej o zadanym maximum

Funkcja tworzy liczby odniesienia, by lepiej dobrac ilosci danych do testu

Parametry

in	*p*	- int, w tablicy ustalane sa argumenty liczb odniesienia
in	*ile-int,jak*	duzo ma zostac stworzonych liczb odniesienia

Warunek wstępny

Ilosc liczb odniesienia musi byc podzielna przez 4

Definicja w linii 293 pliku Benchmark.hh.

4.1.3.4 template<typename T > void Benchmark< T >::Testuj (int *MaxIloscDanych,* void(∗)(T ∗, int) *wsk_fun*)

brief Funkcja mierzy czas trwania funkcji dla okreslonej ilosci danych

Funkcja sluzy do badania zlozonosci obliczeniowej danej funkcji

Parametry

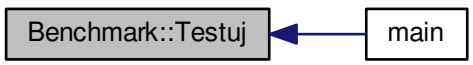
in	*MaxIloscDanych*	- int, maksymalna ilosc danych do testu
in	*wsk_fun=*	wskaznik na funkcje testowana o arg:double∗,int

Warunek wstępny

Funkcja w argumencie musi być typu(TypSzablonuBenchmark,maxLiczbaElementow)
Ilosc cykli ustawia sie w konstruktorze Benchmarku,maxElem=75 000 000

Definicja w linii 247 pliku Benchmark.hh.

Oto graf wywoływań tej funkcji:



4.1.3.5 template<typename T > void Benchmark< T >::TransformacjaBoxa_Mullera (float ∗ *a*)

brief Funkcja zamienia liczby float z rozkladu rownomiernego na Gaussowy

Funkcja zamienia liczby z rozkladu rownomiernego na rozklad Gaussa

Parametry

in	*a[]*	- typ float,wskaznik na tablice 2-elementowa

Warunek wstępny

2 liczby z rozkladu rownomiernego

Definicja w linii 231 pliku Benchmark.hh.

4.1.3.6 template<typename T > void Benchmark< T >::UstalRozmiarTablicyZlozonosciObliczeniowej (int *roz*)

brief Ustalenie rozmiaru tablicy zlozonosci obliczeniowej

Funkcja ustala na nowo rozmiar tablicy ZlozonoscObliczeniowa

Parametry

in	*roz*	- typ int, rozmiar tablicy ZlozonoscObliczeniowa

Warunek wstępny

zmienna roz musi byc dodatnia

Definicja w linii 119 pliku Benchmark.hh.

4.1.3.7 template<typename T > std::ostream & Benchmark< T >::ZapiszWynikiZlozonosciObliczeniowej (std::ostream & *Strm*)

Funkcja sluzy do zapisania wynikow z tablicy ZlozonoscObliczeniowa

Parametry

in	*&Strum*	- referencja do strumienia wyjsciowego
in	*roz*	- ilosc elementow w tablicy ZlozonoscObliczniowa przez 2

Zwraca

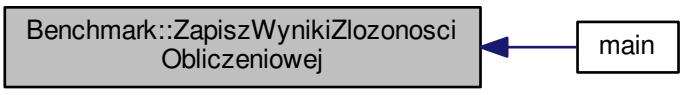
Zwraca referencje do strumienia wyjsciowego

Warunek wstępny

Poprawnie wczytane wartosci tablicy ZlozonoscObliczeniowa

Definicja w linii 313 pliku Benchmark.hh.

Oto graf wywoływań tej funkcji:



4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 template<typename T> unsigned long int Benchmark< T >::IloscDanych

brief ilosc liczb z rozkladu Gaussa

Definicja w linii 30 pliku Benchmark.hh.

4.1.4.2 template<typename T> T* Benchmark< T >::LiczbyGaussowe

brief wskaznik na tablice przechowujaca elementy z rozkladu gaussa

Definicja w linii 26 pliku Benchmark.hh.

4.1.4.3 template<typename T> int Benchmark< T >::rozmiar

brief ilosc liczb zlozonosci obliczeniowej

Definicja w linii 34 pliku Benchmark.hh.

4.1.4.4 template<typename T> unsigned long int** Benchmark< T >::ZlozonoscObliczeniowa

brief wskaznik na tablice przechowujaca wartosci zlozonosci obliczeniowej(ilosc danych,czas)

Definicja w linii 22 pliku Benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **Benchmark.hh**

4.2 Dokumentacja szablonu klasy Kolejka< T >

klasa **Kolejka** (str. 11) sluzy do wykonywania podstawowych operacji na Kolejce: dodaj,odejmij element. Przechowuje informacje o ilosci wszysktich elementow.

`#include <Kolejka.hh>`

Metody publiczne

- **Kolejka** ()
- ~**Kolejka** ()
- int **size** ()
- void **push_front** (T value)
- void **pop_back** ()
- void **show** ()

Atrybuty prywatne

- **Node**< T > * **head**
- int **_size**

4.2.1 Opis szczegółowy

template<typename T>class Kolejka< T >

Definicja w linii 21 pliku Kolejka.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 template<typename T > Kolejka< T >::Kolejka ( )

brief Konstruktor bezparametryczny

Konstruktor bezparametryczny, ustawia parametry na 0

Definicja w linii 59 pliku Kolejka.hh.

4.2.2.2 template<typename T > Kolejka< T >::~Kolejka ( )

Destruktor, usuwa kolejne elementy kolejki zaczynajac od poczatku

Definicja w linii 69 pliku Kolejka.hh.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 template<typename T > void Kolejka< T >::pop_back ( )

brief Funkcja zdejmuje element z konca kolejki

Funkcja usuwa element z konca kolejki

Warunek wstępny

Kolejka (str. 11) nie moze byc pusta

Definicja w linii 106 pliku Kolejka.hh.

4.2.3.2 template<typename T > void Kolejka< T >::push_front (T *value*)

brief Funkcja dodaje element na poczatek kolejki

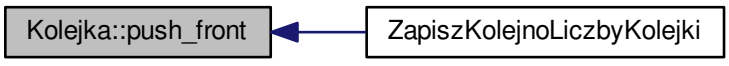
Funkcja sluzy do dodania elementu na poczatek kolejki

Parametry

in	*value-typ*	int, wartosc elementu zmiennej dodanej do kolejki

Definicja w linii 92 pliku Kolejka.hh.

Oto graf wywoływań tej funkcji:



4.2.3.3 template<typename T > void Kolejka< T >::show ( )

brief Funkcja wyswietla wszystkie elementy na standardowe wyjscie

Funkcja wyswietla elementy kolejki

Definicja w linii 134 pliku Kolejka.hh.

4.2.3.4 template<typename T > int Kolejka< T >::size ( )

brief Funkcja zwraca rozmiar kolejki

Zwraca

Funkcja zwraca wartosc rozmiaru kolejki

Definicja w linii 82 pliku Kolejka.hh.

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 template<typename T> int Kolejka< T >::_size [private]

brief Informacja o rozmiarze kolejki

Definicja w linii 30 pliku Kolejka.hh.

4.2.4.2 template<typename T> Node<T>* Kolejka< T >::head [private]

brief wskaznik do ktorego doczepione sa kolejne elementy

Definicja w linii 26 pliku Kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **Kolejka.hh**

4.3 Dokumentacja szablonu klasy List< T >

klasa **List** (str. 13) sluzy do wykonywania podstawowych operacji na Liscie: dodaj,odejmij element. Przechowuje informacje o ilosci wszysktich elementow.

`#include <Lista.hh>`

Metody publiczne

- **List** ()
- ~**List** ()
- int **size** ()
- void **push_front** (T value)
- void **pop_front** ()
- void **push_back** (T value)
- void **pop_back** ()
- void **show** ()
- void **showOdKonca** ()
- void **push** (T value, int nr=0)
- T & **operator[]** (int a)

Atrybuty publiczne

- **NodeL**< T > ∗ **head**
- **NodeL**< T > ∗ **tail**
- int **_size**

4.3.1 Opis szczegółowy

template<**typename T**>**class List**< **T** >

Definicja w linii 25 pliku Lista.hh.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 template<typename T > List< T >::List ( )

brief Konstruktor bezparametryczny

Konstruktor bezparametryczny, ustawia parametry na 0

Definicja w linii 101 pliku Lista.hh.

4.3.2.2 template<typename T > List< T >::~List ( )

brief Destruktor

Destruktor, usuwa kolejne elementy listy zaczynajac od poczatku

Definicja w linii 112 pliku Lista.hh.

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 template<typename T> T& List< T >::operator[] (int *a*) [inline]

Przeciazony operator indeksowania zwraca referencje do elementu o indeksie a

Definicja w linii 87 pliku Lista.hh.

4.3.3.2 template<typename T > void List< T >::pop_back ( )

brief Funkcja zdejmuje element z konca listy

Funkcja usuwa element z konca listy

Warunek wstępny

Lista nie moze byc pusta

Definicja w linii 229 pliku Lista.hh.

4.3.3.3 template<typename T > void List< T >::pop_front ( )

brief Funkcja zdejmuje element z poczatku listy

Funkcja usuwa element z poczatku listy

Warunek wstępny

Lista nie moze byc pusta

Definicja w linii 152 pliku Lista.hh.

4.3.3.4 template<typename T > void List< T >::push (T *value,* int *nr* = 0)

brief Funkcja dodaje element przed elementem o indeksie nr

Funkcja dodaje element przed elementem o indeksie nr

Parametry

in	*value-wybrany*	typ, wartosc elementu dodanego do listy
in	*nr-*	indeks elementu przed ktorym ma byc dodany element

Warunek wstępny

indeksowanie od 0

Definicja w linii 196 pliku Lista.hh.

4.3.3.5 template<typename T > void List< T >::push_back (T *value*)

brief Funkcja dodaje element na koniec listy

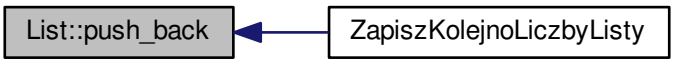
Funkcja dodaje element na koniec listy

Parametry

in	*value*	- typ int, wartosc elementu dodanego na koniec listy

Definicja w linii 171 pliku Lista.hh.

Oto graf wywoływań tej funkcji:



4.3.3.6 template<typename T > void List< T >::push_front (T *value*)

brief Funkcja dodaje element na poczatek listy

Funkcja sluzy do dodania elementu na poczatek listy

Parametry

in	*value-typ*	int, wartosc elementu zmiennej dodanej do listy

Definicja w linii 135 pliku Lista.hh.

Oto graf wywoływań tej funkcji:

4.3.3.7 template<typename T > void List< T >::show ( )

brief Funkcja wyswietla wszystkie elementy na standardowe wyjscie

brief Funkcja pokazujaca na strumieniu std::cout zawartosc listy

Funkcja wyswietla elementy listy

Definicja w linii 259 pliku Lista.hh.

4.3.3.8 template<typename T > void List< T >::showOdKonca ( )

brief Funkcja pokazujaca na strumieniu std::cout zawartosc listy od konca

Funkcja wyswietla elementy listy od konca

Definicja w linii 273 pliku Lista.hh.

4.3.3.9 template<typename T > int List< T >::size ( )

brief Funkcja zwraca rozmiar listy

Zwraca

Funkcja zwraca wartosc rozmiaru listy

Definicja w linii 125 pliku Lista.hh.

4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 template<typename T> int List< T >::_size

brief Informacja o rozmiarze listy

Definicja w linii 39 pliku Lista.hh.

4.3.4.2 template<typename T> NodeL<T>* List< T >::head

brief wskaznik do ktorego doczepione sa kolejne elementy listy

Definicja w linii 31 pliku Lista.hh.

4.3.4.3 template<typename T> NodeL<T>* List< T >::tail

brief wskaznik pokazujacy na koniec listy

Definicja w linii 35 pliku Lista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **Lista.hh**

4.4 Dokumentacja szablonu struktury Node< T >

`#include <Kolejka.hh>`

Atrybuty publiczne

- T **val**
- **Node**< T > * **next**

4.4.1 Opis szczegółowy

template<typename T>struct Node< T >

Definicja w linii 10 pliku Kolejka.hh.

4.4.2 Dokumentacja atrybutów składowych

4.4.2.1 template<typename T> Node<T>* Node< T >::next

Definicja w linii 13 pliku Kolejka.hh.

4.4.2.2 template<typename T> T Node< T >::val

Definicja w linii 12 pliku Kolejka.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- **Kolejka.hh**

4.5 Dokumentacja szablonu struktury NodeL< T >

`#include <Lista.hh>`

Atrybuty publiczne

- T **val**
- **NodeL**< T > * **next**
- **NodeL**< T > * **prev**

4.5.1 Opis szczegółowy

template<typename T>struct NodeL< T >

Definicja w linii 12 pliku Lista.hh.

4.5.2 Dokumentacja atrybutów składowych

4.5.2.1 template<typename T> NodeL<T>* NodeL< T >::next

Definicja w linii 15 pliku Lista.hh.

4.5.2.2 template<typename T> NodeL<T>* NodeL< T >::prev

Definicja w linii 16 pliku Lista.hh.

4.5.2.3 template<typename T> T NodeL< T >::val

Definicja w linii 14 pliku Lista.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- **Lista.hh**

4.6 Dokumentacja szablonu klasy Stack< T >

`#include <Stack.hh>`

Metody publiczne

- **Stack** (int **capacity**=10)
- void **push** (T value)
- T **peek** ()
- int **size** ()
- ~**Stack** ()
- void **pop** ()
- T & **operator[]** (int a)

Atrybuty publiczne

- T * **top**
- int **capacity**
- T * **storage**

4.6.1 Opis szczegółowy

template<typename T>class Stack< T >

Definicja w linii 11 pliku Stack.hh.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 template<typename T > Stack< T >::Stack (int *capacity* = 10)

Konstruktor parametryczny klasy **Stack** (str. 18)

Parametry

in	*capacity*	- typ int, rozmiar stosu

Definicja w linii 66 pliku Stack.hh.

4.6.2.2 template<typename T > Stack< T >::~Stack ( )

Destruktor klasy **Stack** (str. 18)

Definicja w linii 125 pliku Stack.hh.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 template<typename T> T& Stack< T >::operator[] (int *a*) [inline]

brief Przeciazony operator indeksowania, umozliwia traktowanie stosu jak tablicy

Definicja w linii 54 pliku Stack.hh.

4.6.3.2 template<typename T > T Stack< T >::peek ( )

Funkcja pokazuje element znajdujacy sie na szczycie stosu

Warunek wstępny

Stos nie moze byc pusty

Definicja w linii 104 pliku Stack.hh.

4.6.3.3 template<typename T > void Stack< T >::pop ( )

Funkcja pop zdejmuje ostatni element ze stosu

Warunek wstępny

Stos nie moze byc pusty

Definicja w linii 136 pliku Stack.hh.

4.6.3.4 template<typename T > void Stack< T >::push (T *value*)

Funkcja dodaje element na koniec tablicy stosu

Parametry

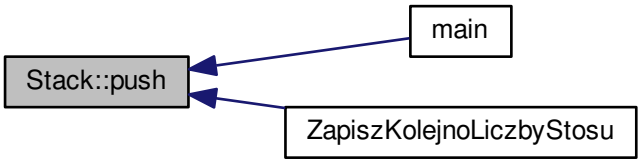
in	*value*	- typ int, wartosc dodana do stosu

Warunek końcowy

wykorzystana metoda podwajania do powiekszania stosu

Definicja w linii 83 pliku Stack.hh.

Oto graf wywoływań tej funkcji:



4.6.3.5 template<typename T > int Stack< T >::size ( )

Funkcja pokazuje ilosc elementow stosu

Zwraca

zwraca ilosc elementow stosu

Definicja w linii 116 pliku Stack.hh.

4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 template<typename T> int Stack< T >::capacity

Definicja w linii 21 pliku Stack.hh.

4.6.4.2 template<typename T> T* Stack< T >::storage

Definicja w linii 25 pliku Stack.hh.

4.6.4.3 template<typename T> T* Stack< T >::top

Definicja w linii 17 pliku Stack.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- **Stack.hh**

Rozdział 5

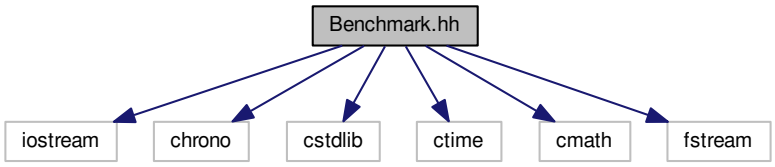
Dokumentacja plików

5.1 Dokumentacja pliku Benchmark.hh

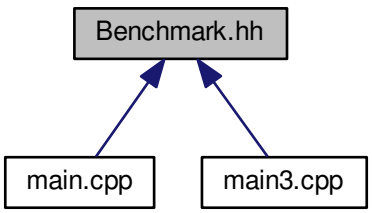
Klasa **Benchmark** (str. 7) sluzy do przechowywania wynikow zlozonosci obliczeniowej i danych wejsciowych,generowania liczb rozkladu Gaussowego.

```
#include <iostream>
#include <chrono>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <fstream>
```

Wykres zależności załączania dla Benchmark.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class **Benchmark**< **T** >

Funkcje

- template<typename T >
  std::ostream & **operator**<< (std::ostream &Strm, const **Benchmark**< T > &ben)
- template<typename T >
  std::istream & **operator**>> (std::istream &Strm, **Benchmark**< T > &ben)

5.1.1 Dokumentacja funkcji

5.1.1.1 template<typename T > std::ostream& operator<< (std::ostream & *Strm,* const Benchmark< T > & *ben*)

Funkcja operatorowa pozwala na wypisanie wszystkich liczb tablicy LiczbyGaussowe

Parametry

in,out	*&Strm*	- referencja do strumienia wyjsciowego
in,out	*&ben*	refencja do klasy typu **Benchmark** (str. 7)

Zwraca

Zwraca referencje do strumienia wyjsciowego

Warunek wstępny

Poprawne wczytanie liczb tablicy LiczbyGaussowe

Definicja w linii 140 pliku Benchmark.hh.

5.1.1.2 template<typename T > std::istream& operator>> (std::istream & *Strm,* Benchmark< T > & *ben*)

Funkcja operatorowa pozwala na wczytanie liczby typu double do tablicy LiczbyGaussowe

Parametry

in,out	*&Strm*	- referencja do strumienia wejsciowego
in,out	*&ben*	referencja do klasy typu **Benchmark** (str. 7)

Zwraca

Zwraca referencje do strumienia wejsciowego

Warunek wstępny

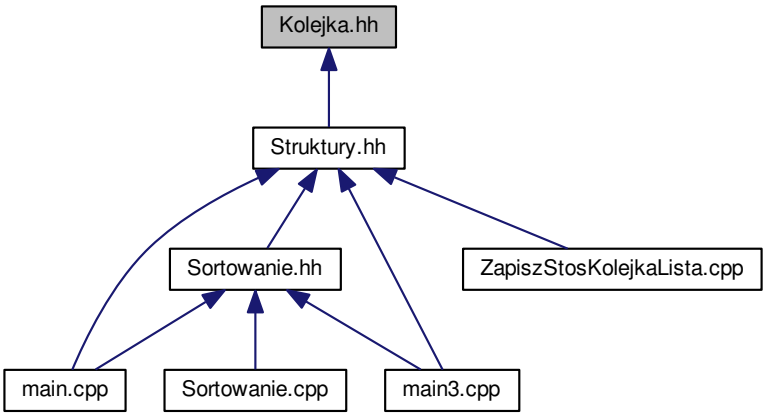
Liczba tylko typu double

Definicja w linii 158 pliku Benchmark.hh.

5.2 Dokumentacja pliku Kolejka.hh

Struktura przechowujaca wartosc wezla i wskaznik na nastepny element typu **Node** (str. 17).

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- struct **Node**< **T** >
- class **Kolejka**< **T** >

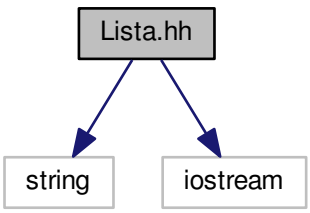
  *klasa **Kolejka** (*str. *11) sluzy do wykonywania podstawowych operacji na Kolejce: dodaj,odejmij element. Przechowuje informacje o ilosci wszysktich elementow.*

5.3 Dokumentacja pliku Lista.hh

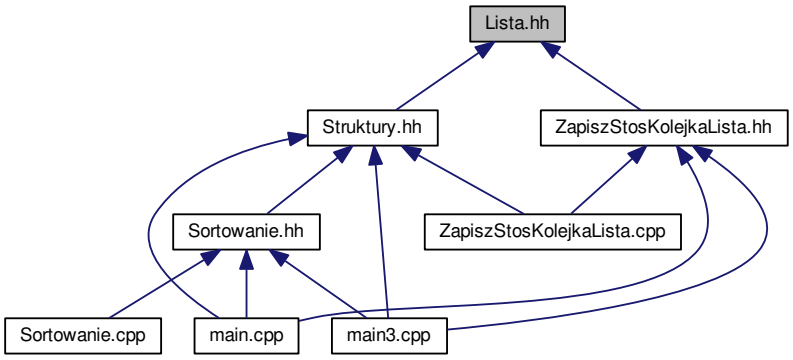
Struktura przechowujaca wartosc wezla i wskaznik na nastepny element typu **Node** (str. 17).

```
#include <string>
#include <iostream>
```

Wykres zależności załączania dla Lista.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

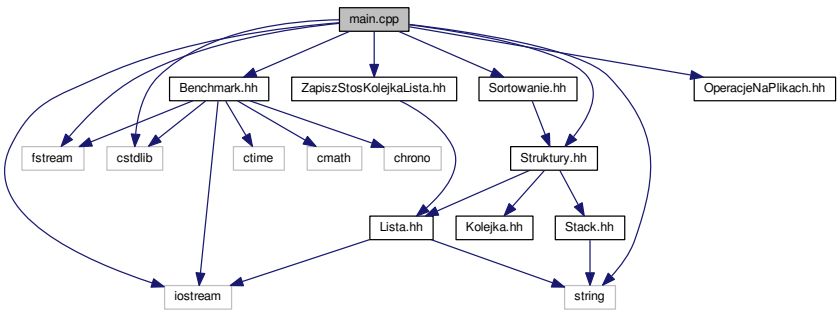
- struct **NodeL**< **T** >
- class **List**< **T** >

  *klasa **List** (str. 13) sluzy do wykonywania podstawowych operacji na Liscie: dodaj,odejmij element. Przechowuje informacje o ilosci wszysktich elementow.*

5.4 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include "Benchmark.hh"
#include "ZapiszStosKolejkaLista.hh"
#include "OperacjeNaPlikach.hh"
#include "Struktury.hh"
#include "Sortowanie.hh"
```

Wykres zależności załączania dla main.cpp:



Funkcje

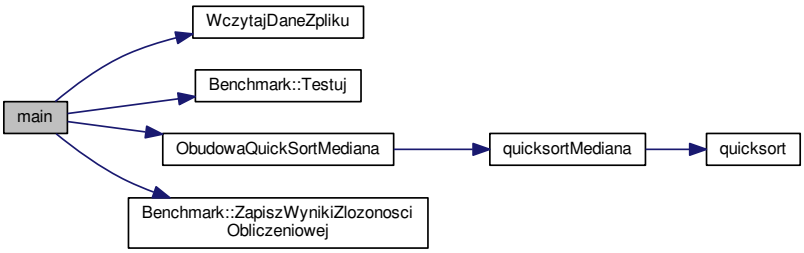
- int **main** (int argc, char ∗argv[])

5.4.1 Dokumentacja funkcji

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

Definicja w linii 13 pliku main.cpp.

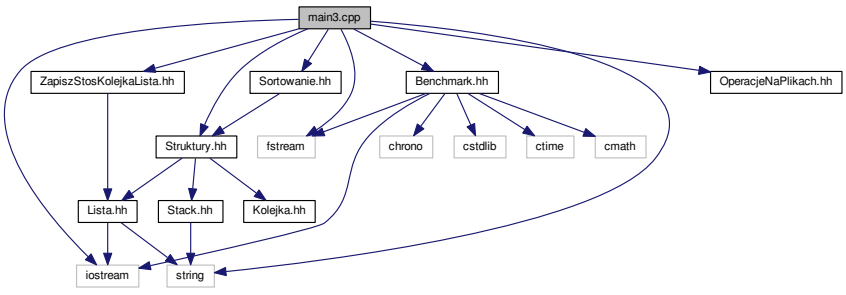
Oto graf wywołań dla tej funkcji:



5.5 Dokumentacja pliku main3.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "Benchmark.hh"
#include "ZapiszStosKolejkaLista.hh"
#include "OperacjeNaPlikach.hh"
#include "Struktury.hh"
#include "Sortowanie.hh"
```

Wykres zależności załączania dla main3.cpp:



Funkcje

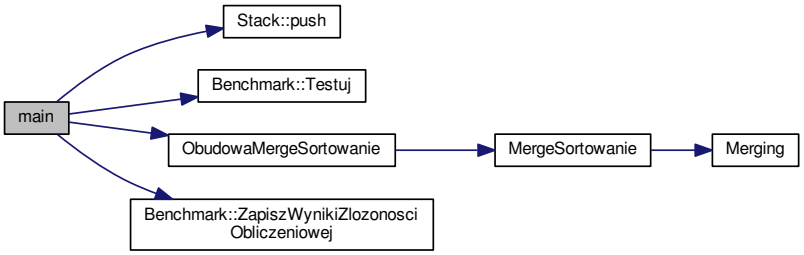
- int **main** (int argc, char ∗argv[])

5.5.1 Dokumentacja funkcji

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

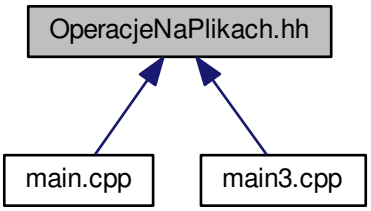
Definicja w linii 12 pliku main3.cpp.

Oto graf wywołań dla tej funkcji:



5.6 Dokumentacja pliku OperacjeNaPlikach.hh

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- template<typename T >
  std::istream & **WczytajDaneZpliku** (std::istream &Strm, unsigned long int IloscDanych, T ∗dane)

5.6.1 Dokumentacja funkcji

5.6.1.1 template<typename T > std::istream& WczytajDaneZpliku (std::istream & *Strm,* unsigned long int *IloscDanych,* T ∗ *dane*)

Funkcja sluzy do wczytania elementow ze strumienia

Parametry

in	*&Strm-*	referencja do strumienia wejsciowego
	IloscDanych	- ilosc, jak wiele danych ma byc wczytane
	∗dane	- wskaznik na strukture do ktorej beda wczytywane dane

Zwraca

zwraca referencje do strumienia wejsciowego

Definicja w linii 11 pliku OperacjeNaPlikach.hh.

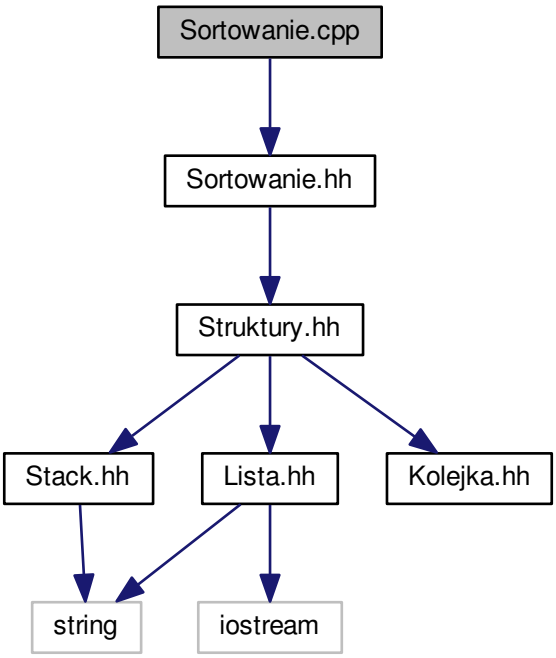
Oto graf wywoływań tej funkcji:

5.7 Dokumentacja pliku README.md

5.8 Dokumentacja pliku Sortowanie.cpp

```
#include "Sortowanie.hh"
```

Wykres zależności załączania dla Sortowanie.cpp:



Funkcje

- void **quicksort** (int ∗tablica, int lewy, int prawy)
- void **quicksortMediana** (int ∗tablica, int lewy, int prawy)
- void **ObudowaQuickSort** (int ∗tablica, int rozmiar)
- void **ObudowaQuickSortMediana** (int ∗tablica, int rozmiar)
- void **quicksortLista** (**List**< int > ∗tablica, **NodeL**< int > ∗lewy, **NodeL**< int > ∗prawy, int indexLewy, int indexPrawy)
- void **Ob** (**List**< int > ∗list, int rozmiar)
- void **ObudowaMergeSortowanie** (**Stack**< int > ∗tab, int rozmiar)
- void **Merging** (**Stack**< int > ∗tab, int lewy, int srodek, int prawy)
- void **MergeSortowanie** (**Stack**< int > ∗tab, int lewy, int prawy)

5.8.1 Dokumentacja funkcji

5.8.1.1 void MergeSortowanie (Stack< int > ∗ *tab,* int *lewy,* int *prawy*)

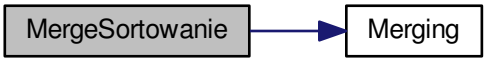
brief Funkcja sortuje tablice skladajaca sie z elementow typu int za pomoca algorytmu Scalania

Parametry

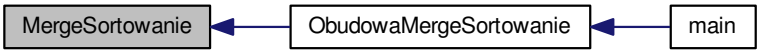
]	tab- typ Stack<int> ∗, wskaznik na tablice do posortowania
]	lewy - typ int, indeks poczatku lewej podtablicy
]	prawy - typ int, indeks konca prawej podtablicy

Definicja w linii 187 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



5.8.1.2 void Merging (Stack< int > ∗ *tab,* int *lewy,* int *srodek,* int *prawy*)

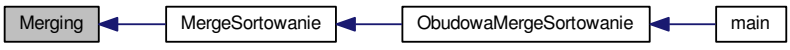
brief Funkcja scala dwa zbiory liczb rosnaca, jest funkcja skladowa MergeSortowanie

Parametry

]	tab- typ Stack<int> ∗, wskaznik na tablice do scalenia
]	lewy - typ int, indeks poczatku lewej podtablicy
]	srodek -typ int, indeks konca lewj podtablicy
]	prawy - typ int, indeks konca prawej podtablicy

Definicja w linii 160 pliku Sortowanie.cpp.

Oto graf wywoływań tej funkcji:



5.8.1.3 void Ob (List< int > ∗ *list,* int *rozmiar*)

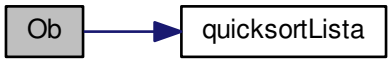
brief Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Parametry

]	list - typ List<int>∗, wskaznik na liste do posortowania
]	rozmiar - typ int, ilosc elementow do posortowania zaczynajac od poczatku listy

Definicja w linii 137 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



5.8.1.4 void ObudowaMergeSortowanie (Stack< int > ∗ *tab,* int *rozmiar*)

brief Funkcja obudowuje funkcje MergeSortowanie,zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Definicja w linii 146 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

5.8.1.5 void ObudowaQuickSort (int ∗ *tablica,* int *rozmiar*)

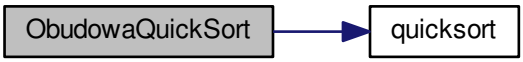
Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Parametry

]	tablica - typ int∗, wskaznik na tablice do posortowania
]	rozmiar - typ int, ilosc elementow do posortowania zaczynajac od indeksu 0

Definicja w linii 81 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



5.8.1.6 void ObudowaQuickSortMediana (int ∗ *tablica,* int *rozmiar*)

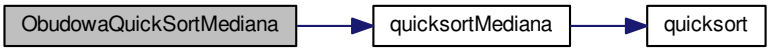
Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Parametry

]	tablica - typ int∗, wskaznik na tablice do posortowania
]	rozmiar - typ int, ilosc elementow do posortowania zaczynajac od indeksu 0

Definicja w linii 93 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

5.8.1.7 void quicksort (int ∗ *tablica,* int *lewy,* int *prawy*)

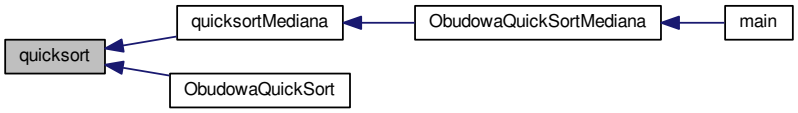
Funkcja sortuje tablice typu int za pomoca algorytmu quicksort

Parametry

]	tablica - typ int∗, wskaznik na tablice do posortowania
]	lewy - typ int, lewy indeks tablicy
]	prawy - typ int, prawy indeks tablicy

Definicja w linii 9 pliku Sortowanie.cpp.

Oto graf wywoływań tej funkcji:



5.8.1.8 void quicksortLista (List< int > ∗ *tablica,* NodeL< int > ∗ *lewy,* NodeL< int > ∗ *prawy,* int *indexLewy,* int *indexPrawy*)

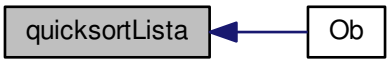
Funkcja sortuje Liste skladajaca sie z elementow typu int za pomoca algorytmu quicksort

Parametry

]	tablica - typ List<int>∗, wskaznik na Liste do posortowania
]	lewy - NodeL<int> ∗ ,wskaznik na lewy wezel Listy
]	prawy - NodeL<int> ∗, wskaznik na prawy wezel Listy

Definicja w linii 106 pliku Sortowanie.cpp.

Oto graf wywoływań tej funkcji:



5.8.1.9 void quicksortMediana (int ∗ *tablica,* int *lewy,* int *prawy*)

Funkcja sortuje tablice typu int za pomoca algorytmu quicksort z wyborem pivota na podstawie mediany 3 elementow

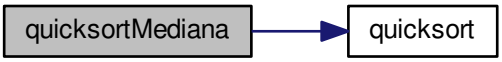
Parametry

]	tablica - typ int∗, wskaznik na tablice do posortowania
]	lewy - typ int, lewy indeks tablicy

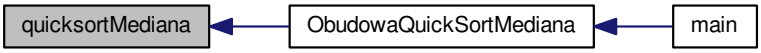
]	prawy - typ int, prawy indeks tablicy

Definicja w linii 41 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



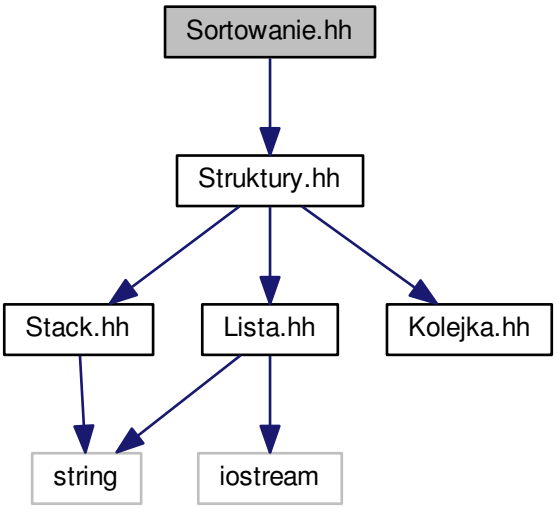
Oto graf wywoływań tej funkcji:



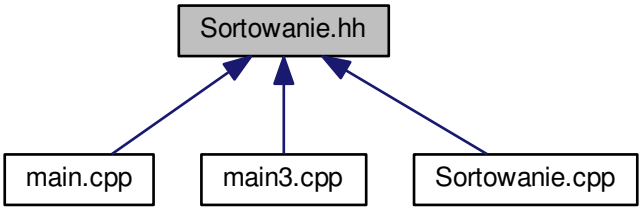
5.9 Dokumentacja pliku Sortowanie.hh

```
#include "Struktury.hh"
```

Wykres zależności załączania dla Sortowanie.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- void **quicksort** (int ∗tablica, int lewy, int prawy)
- void **ObudowaQuickSort** (int ∗tablica, int rozmiar)
- void **quicksortLista** (**List**< int > ∗tablica, **NodeL**< int > ∗lewy, **NodeL**< int > ∗prawy, int l, int p)
- void **Ob** (**List**< int > ∗tablica, int rozmiar)
- void **quicksortMediana** (int ∗tablica, int lewy, int prawy)
- void **ObudowaQuickSortMediana** (int ∗tablica, int rozmiar)
- void **MergeSortowanie** (**Stack**< int > ∗tablica, int lewy, int prawy)
- void **Merging** (**Stack**< int > ∗tablica, int lewy, int srodkowy, int prawy)
- void **ObudowaMergeSortowanie** (**Stack**< int > ∗tab, int rozmiar)

5.9.1 Dokumentacja funkcji

5.9.1.1 void MergeSortowanie (Stack< int > ∗ *tab,* int *lewy,* int *prawy*)

brief Funkcja sortuje tablice skladajaca sie z elementow typu int za pomoca algorytmu Scalania

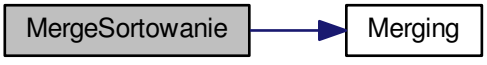
brief Funkcja sortuje tablice skladajaca sie z elementow typu int za pomoca algorytmu Scalania

Parametry

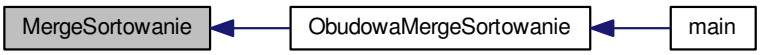
]	tab- typ Stack<int> ∗, wskaznik na tablice do posortowania
]	lewy - typ int, indeks poczatku lewej podtablicy
]	prawy - typ int, indeks konca prawej podtablicy

Definicja w linii 187 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



5.9.1.2 void Merging (Stack< int > * *tab,* int *lewy,* int *srodek,* int *prawy*)

brief Funkcja scala dwa zbiory liczb, jest funkcja skladowa MergeSortowanie

brief Funkcja scala dwa zbiory liczb rosnaca, jest funkcja skladowa MergeSortowanie

Parametry

]	tab- typ Stack<int> *, wskaznik na tablice do scalenia
]	lewy - typ int, indeks poczatku lewej podtablicy
]	srodek -typ int, indeks konca lewj podtablicy
]	prawy - typ int, indeks konca prawej podtablicy

Definicja w linii 160 pliku Sortowanie.cpp.

Oto graf wywoływań tej funkcji:

5.9.1.3 void Ob (List< int > * *list,* int *rozmiar*)

brief Funkcja obudowuje funkcje quicksortLista,zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

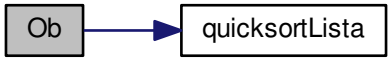
brief Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Parametry

]	list - typ List<int>*, wskaznik na liste do posortowania
]	rozmiar - typ int, ilosc elementow do posortowania zaczynajac od poczatku listy

Definicja w linii 137 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



5.9.1.4 void ObudowaMergeSortowanie (Stack< int > * *tab,* int *rozmiar*)

brief Funkcja obudowuje funkcje MergeSortowanie,zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Definicja w linii 146 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

5.9.1.5 void ObudowaQuickSort (int * *tablica,* int *rozmiar*)

brief Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

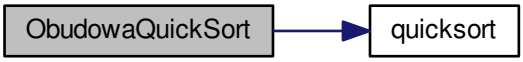
Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Parametry

]	tablica - typ int*, wskaznik na tablice do posortowania
]	rozmiar - typ int, ilosc elementow do posortowania zaczynajac od indeksu 0

Definicja w linii 81 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



5.9.1.6 void ObudowaQuickSortMediana (int * *tablica,* int *rozmiar*)

brief Funkcja obudowuje funkcje quicksortMediana,zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

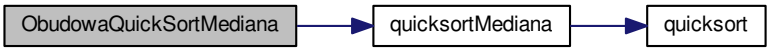
Funkcja obudowuje funkcje quicksort, zeby mogla zostac wczytana przez klase **Benchmark** (str. 7)

Parametry

]	tablica - typ int∗, wskaznik na tablice do posortowania
]	rozmiar - typ int, ilosc elementow do posortowania zaczynajac od indeksu 0

Definicja w linii 93 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

5.9.1.7 void quicksort (int ∗ *tablica,* int *lewy,* int *prawy*)

brief Funkcja sortuje tablice typu int za pomoca algorytmu quicksort

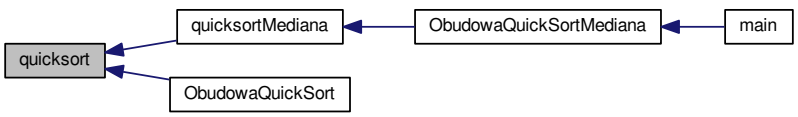
Funkcja sortuje tablice typu int za pomoca algorytmu quicksort

Parametry

]	tablica - typ int∗, wskaznik na tablice do posortowania
]	lewy - typ int, lewy indeks tablicy
]	prawy - typ int, prawy indeks tablicy

Definicja w linii 9 pliku Sortowanie.cpp.

Oto graf wywoływań tej funkcji:



5.9.1.8 void quicksortLista (List< int > * *tablica,* NodeL< int > * *lewy,* NodeL< int > * *prawy,* int *indexLewy,* int *indexPrawy*)

brief Funkcja sortuje Liste skladajace sie z elementow typu int za pomoca algorytmu quicksort

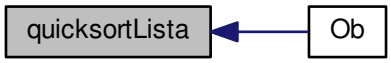
Funkcja sortuje Liste skladajaca sie z elementow typu int za pomoca algorytmu quicksort

Parametry

]	tablica - typ List<int>*, wskaznik na Liste do posortowania
]	lewy - NodeL<int> * ,wskaznik na lewy wezel Listy
]	prawy - NodeL<int> *, wskaznik na prawy wezel Listy

Definicja w linii 106 pliku Sortowanie.cpp.

Oto graf wywoływań tej funkcji:



5.9.1.9 void quicksortMediana (int * *tablica,* int *lewy,* int *prawy*)

brief Sortowanie za pomoca QuickSorta z wybranem pivotem na podstawie mediany

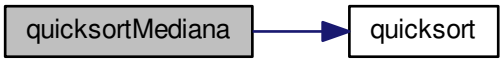
Funkcja sortuje tablice typu int za pomoca algorytmu quicksort z wyborem pivota na podstawie mediany 3 elementow

Parametry

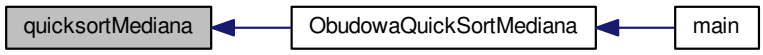
]	tablica - typ int*, wskaznik na tablice do posortowania
]	lewy - typ int, lewy indeks tablicy
]	prawy - typ int, prawy indeks tablicy

Definicja w linii 41 pliku Sortowanie.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

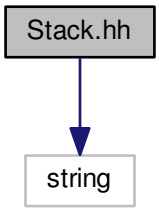


5.10 Dokumentacja pliku Stack.hh

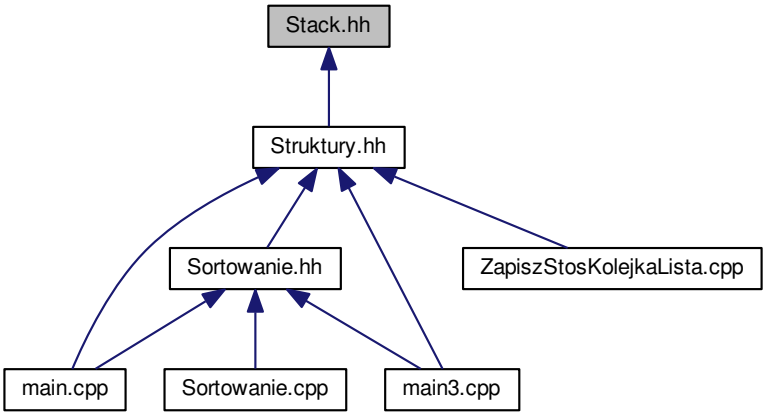
Klasa **Stack** (str. 18) sluzy do przechowywania, dodawania,zdejmowania kolejnych elementow stosu.

```
#include <string>
```

Wykres zależności załączania dla Stack.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class **Stack**< **T** >

Funkcje

- template<typename T >
  std::ostream & **operator**<< (std::ostream &out, const **Stack**< T > &stack)

5.10.1 Dokumentacja funkcji

5.10.1.1 template<typename T > std::ostream& operator<< (std::ostream & *out,* const Stack< T > & *stack*)

Funkcja operatorowa sluzy do wyswietlania stosu,zbedna

Parametry

in	*&out*	- referencja do strumienia wyjsciowego
in	*&stack-*	referencja do stosu

Zwraca

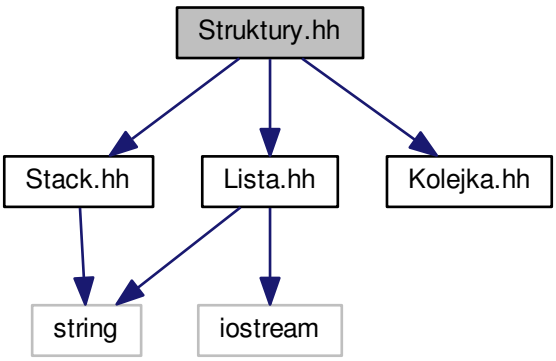
zwraca referencje do strumienia wyjsciowego

Definicja w linii 151 pliku Stack.hh.

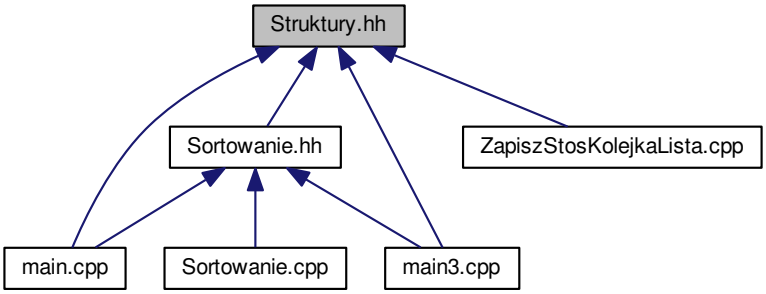
5.11 Dokumentacja pliku Struktury.hh

```
#include "Stack.hh"
#include "Lista.hh"
#include "Kolejka.hh"
```

Wykres zależności załączania dla Struktury.hh:



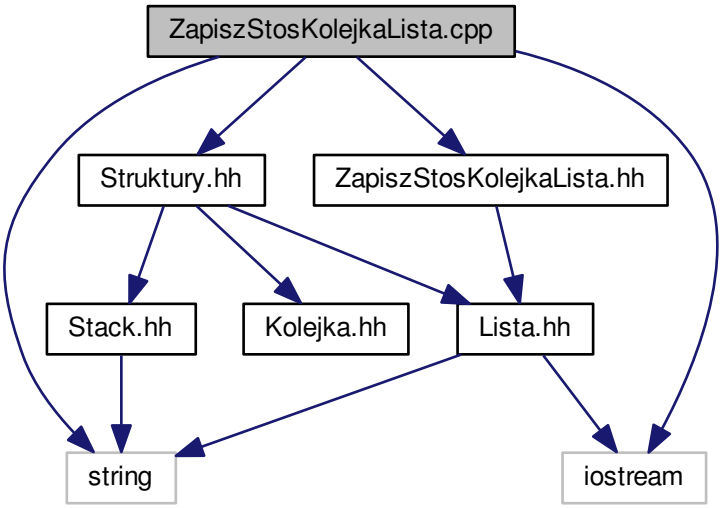
Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



5.12 Dokumentacja pliku ZapiszStosKolejkaLista.cpp

```
#include <string>
#include <iostream>
#include "Struktury.hh"
#include "ZapiszStosKolejkaLista.hh"
```

Wykres zależności załączania dla ZapiszStosKolejkaLista.cpp:



Funkcje

- void **ZapiszKolejnoLiczbyStosu** (double ∗Gausowe, int size)
- void **ZapiszKolejnoLiczbyListy** (double ∗Gausowe, int size)
- void **ZapiszKolejnoLiczbyKolejki** (double ∗Gausowe, int size)

- void **WczytajListe** (std::istream &Strm, unsigned long int IloscDanych, **List**< int > *dane)

5.12.1 Dokumentacja funkcji

5.12.1.1 void WczytajListe (std::istream & *Strm,* unsigned long int *IloscDanych,* List< int > * *dane*)

brief Funkcja sluzy do Wczytania Listy o elementach typu int

Parametry

]	Strm - referencja do strumienia wejsciowego
]	IloscDanych - typ int, oznacza jak wiele danych ma byc wczytane
]	dane - typ List<int>*, lista gdzie beda wczytane dane

Definicja w linii 50 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:

5.12.1.2 void ZapiszKolejnoLiczbyKolejki (double * *Gausowe,* int *size*)

Funkcja sluzy do zapisywania danych na Kolejce utworzonej w tej funkcji

Parametry

]	Gausowe - typ double *, wskaznik na tablice typu double
]	size - rozmiar, jak wiele liczb ma byc zapisane

Definicja w linii 37 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:

5.12.1.3 void ZapiszKolejnoLiczbyListy (double * *Gausowe,* int *size*)

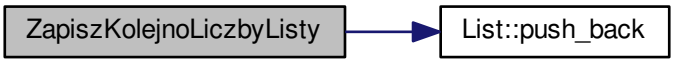
Funkcja sluzy do zapisywania danych na liscie utworzonej w tej funkcji

Parametry

]	Gausowe - typ double ∗, wskaznik na tablice typu double
]	size - rozmiar, jak wiele liczb ma byc zapisane

Definicja w linii 25 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:



5.12.1.4 void ZapiszKolejnoLiczbyStosu (double ∗ *Gausowe,* int *size*)

Funkcja sluzy do zapisywania danych na stosie utworzonym w tej funkcji

Parametry

]	Gausowe - typ double ∗, wskaznik na tablice typu double
]	size - rozmiar, jak wiele liczb ma byc zapisane

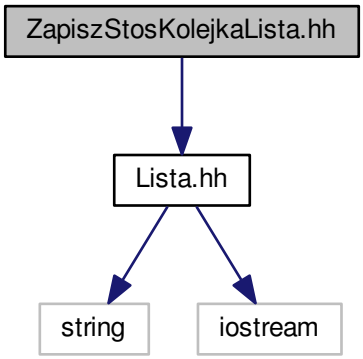
Definicja w linii 13 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:

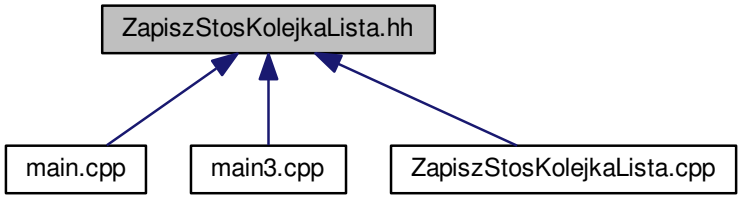
5.13 Dokumentacja pliku ZapiszStosKolejkaLista.hh

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

Wykres zależności załączania dla ZapiszStosKolejkaLista.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- void **ZapiszKolejnoLiczbyStosu** (double ∗Gausowe, int size)
- void **ZapiszKolejnoLiczbyListy** (double ∗Gausowe, int size)
- void **ZapiszKolejnoLiczbyKolejki** (double ∗Gausowe, int size)
- void **WczytajListe** (std::istream &Strm, unsigned long int IloscDanych, **List**< int > ∗dane)

5.13.1 Dokumentacja funkcji

5.13.1.1 void WczytajListe (std::istream & *Strm,* unsigned long int *IloscDanych,* List< int > ∗ *dane*)

brief Funkcja sluzy do Wczytania Listy o elementach typu int

brief Funkcja sluzy do Wczytania Listy o elementach typu int

Parametry

]	Strm - referencja do strumienia wejsciowego
]	IloscDanych - typ int, oznacza jak wiele danych ma byc wczytane
]	dane - typ List<int>∗, lista gdzie beda wczytane dane

Definicja w linii 50 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:

5.13.1.2 void ZapiszKolejnoLiczbyKolejki (double ∗ *Gausowe,* int *size*)

brief Funkcja sluzy do zapisywania danych na Kolejce utworzonej w tej funkcji

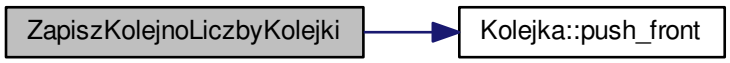
Funkcja sluzy do zapisywania danych na Kolejce utworzonej w tej funkcji

Parametry

]	Gausowe - typ double ∗, wskaznik na tablice typu double
]	size - rozmiar, jak wiele liczb ma byc zapisane

Definicja w linii 37 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:



5.13.1.3 void ZapiszKolejnoLiczbyListy (double ∗ *Gausowe,* int *size*)

brief Funkcja sluzy do zapisywania danych na liscie utworzonej w tej funkcji

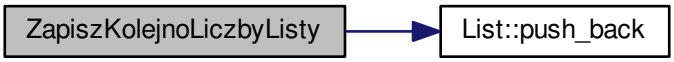
Funkcja sluzy do zapisywania danych na liscie utworzonej w tej funkcji

Parametry

]	Gausowe - typ double ∗, wskaznik na tablice typu double
]	size - rozmiar, jak wiele liczb ma byc zapisane

Definicja w linii 25 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji:



5.13.1.4 void ZapiszKolejnoLiczbyStosu (double * *Gausowe,* int *size*)

brief Funkcja sluzy do zapisywania danych na stosie utworzonym w tej funkcji

Funkcja sluzy do zapisywania danych na stosie utworzonym w tej funkcji

Parametry

]	Gausowe - typ double *, wskaznik na tablice typu double
]	size - rozmiar, jak wiele liczb ma byc zapisane

Definicja w linii 13 pliku ZapiszStosKolejkaLista.cpp.

Oto graf wywołań dla tej funkcji: