

Podstawowe struktury danych

1

Wygenerowano przez Doxygen 1.8.6

Cz, 26 mar 2015 02:14:48

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja klasy benchmark	7
4.1.1 Opis szczegółowy	7
4.1.2 Dokumentacja funkcji składowych	7
4.1.2.1 analityze	8
4.1.2.2 test	8
4.2 Dokumentacja klasy list	9
4.2.1 Opis szczegółowy	10
4.2.2 Dokumentacja konstruktora i destruktora	10
4.2.2.1 list	10
4.2.2.2 ~list	10
4.2.3 Dokumentacja funkcji składowych	11
4.2.3.1 pop	11
4.2.3.2 push	12
4.2.3.3 size	13
4.2.3.4 test	13
4.2.4 Dokumentacja atrybutów składowych	14
4.2.4.1 head	14
4.2.4.2 tail	14
4.3 Dokumentacja klasy list_array	14
4.3.1 Opis szczegółowy	15
4.3.2 Dokumentacja konstruktora i destruktora	15
4.3.2.1 list_array	15

4.3.2.2	<code>~list_array</code>	15
4.3.3	Dokumentacja funkcji składowych	15
4.3.3.1	<code>pop</code>	15
4.3.3.2	<code>push</code>	16
4.3.3.3	<code>size</code>	17
4.3.3.4	<code>test</code>	17
4.3.4	Dokumentacja atrybutów składowych	17
4.3.4.1	<code>n</code>	17
4.3.4.2	<code>temp</code>	17
4.3.4.3	<code>tmp</code>	17
4.4	Dokumentacja struktury <code>node</code>	18
4.4.1	Opis szczegółowy	18
4.4.2	Dokumentacja konstruktora i destruktor	18
4.4.2.1	<code>node</code>	18
4.4.3	Dokumentacja atrybutów składowych	18
4.4.3.1	<code>data</code>	18
4.4.3.2	<code>next</code>	19
4.5	Dokumentacja klasy <code>queue</code>	19
4.5.1	Opis szczegółowy	20
4.5.2	Dokumentacja konstruktora i destruktor	20
4.5.2.1	<code>queue</code>	20
4.5.2.2	<code>~queue</code>	20
4.5.3	Dokumentacja funkcji składowych	20
4.5.3.1	<code>pop</code>	20
4.5.3.2	<code>push</code>	21
4.5.3.3	<code>size</code>	21
4.5.3.4	<code>test</code>	22
4.5.4	Dokumentacja atrybutów składowych	22
4.5.4.1	<code>head</code>	22
4.5.4.2	<code>tail</code>	22
4.6	Dokumentacja klasy <code>stack</code>	22
4.6.1	Opis szczegółowy	24
4.6.2	Dokumentacja konstruktora i destruktor	24
4.6.2.1	<code>stack</code>	24
4.6.2.2	<code>~stack</code>	24
4.6.3	Dokumentacja funkcji składowych	24
4.6.3.1	<code>pop</code>	24
4.6.3.2	<code>push</code>	25
4.6.3.3	<code>size</code>	25
4.6.3.4	<code>test</code>	26

4.6.4	Dokumentacja atrybutów składowych	27
4.6.4.1	head	27
5	Dokumentacja plików	29
5.1	Dokumentacja pliku benchmark.cpp	29
5.2	Dokumentacja pliku benchmark.hh	29
5.3	Dokumentacja pliku generator.cpp	30
5.3.1	Dokumentacja funkcji	31
5.3.1.1	data_generator	31
5.4	Dokumentacja pliku generator.hh	31
5.4.1	Dokumentacja funkcji	32
5.4.1.1	data_generator	32
5.5	Dokumentacja pliku list.cpp	32
5.6	Dokumentacja pliku list.hh	33
5.7	Dokumentacja pliku list_array.cpp	34
5.8	Dokumentacja pliku list_array.hh	35
5.9	Dokumentacja pliku main.cpp	35
5.9.1	Dokumentacja funkcji	36
5.9.1.1	main	36
5.10	Dokumentacja pliku queue.cpp	36
5.11	Dokumentacja pliku queue.hh	37
5.12	Dokumentacja pliku stack.cpp	38
5.13	Dokumentacja pliku stack.hh	39
Indeks		41

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

benchmark	7
list	9
list_array	14
queue	19
stack	22
node	18

Rozdział 2

Indeks klas

2.1 Lista klas

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

benchmark	7
list	9
list_array	14
node	18
queue	19
stack	22

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

benchmark.cpp	Deklaracja funkcji z klasy Benchmark	29
benchmark.hh	Definicja klasy Benchmark	29
generator.cpp	Deklaracja funkcji generującej liczby losowe	30
generator.hh	Definicja generatora liczb losowych	31
list.cpp	Deklaracja klasy list	32
list.hh	Definicja klasy lista	33
list_array.cpp	Deklaracja klasy list	34
list_array.hh	Definicja klasy list_array	35
main.cpp	35
queue.cpp	Deklaracja klasy queue	36
queue.hh	Definicja klasy stack	37
stack.cpp	Deklaracja klasy stack	38
stack.hh	Definicja klasy stack	39

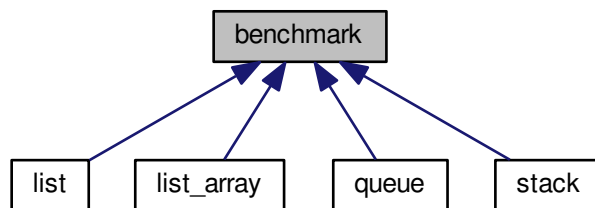
Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy benchmark

```
#include <benchmark.hh>
```

Diagram dziedziczenia dla benchmark



Metody publiczne

- void [analyze](#) (const char *name_output, int repeat, int data_amount)

Metoda analyze zlicza czas wykonywania funkcji [test\(\)](#)

Metody prywatne

- virtual void [test](#) (unsigned long int length)=0

Metoda test funkcja wirtualna , której czas działania ma być aproksymowany przez metoda [analyze\(\)](#)

4.1.1 Opis szczegółowy

Definicja w linii 11 pliku benchmark.hh.

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 void benchmark::analize (const char * name_output, int repeat, int data_amount)

Metoda analize zlicza czas funkcji [test\(\)](#)

Przykład wywołania funkcji :

analize("Plik_wynikowy",100,7) -> Przeprowadza analize czasu trwania funkcji [test\(\)](#) dla 1 miliona danych (ilość danych należy podać jako potęgę 10) , każdy czas trwania funkcji jest ustalany na podstawie średniej arytmetycznej ze 100 prób , wyniki zapisuje do pliku o nazwie Plik_wynikowy.

Parametry

in	<i>name_output</i>	- nazwa pliku wynikowego
in	<i>repeat</i>	- ilość powtórzeń testu
in	<i>data_amount</i>	- ilość wynikowych danych podawana jako potęga liczby 10

Zwraca

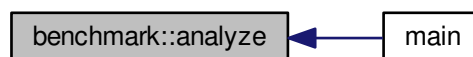
plik.dat z czasami poszczególnych pomiarów oraz ilość testowanych danych.

Definicja w linii 14 pliku benchmark.cpp.

Oto graf wywołań dla tej funkcji:



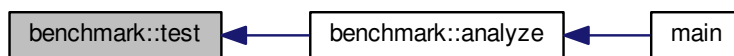
Oto graf wywoływań tej funkcji:



4.1.2.2 virtual void benchmark::test (unsigned long int length) [private],[pure virtual]

Implementowany w [list](#), [list_array](#), [stack](#) i [queue](#).

Oto graf wywoływań tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

- [benchmark.hh](#)
- [benchmark.cpp](#)

4.2 Dokumentacja klasy list

```
#include <list.hh>
```

Diagram dziedziczenia dla list

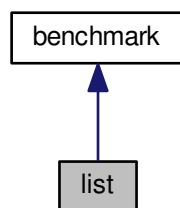
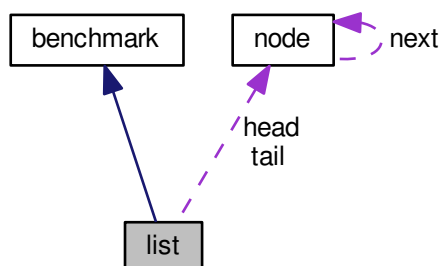


Diagram współpracy dla list:



Metody publiczne

- `list()`
Konstruktor inicjalizujący pustą listę, początek i koniec listy są domyślnie ustawione na NULL.
- `~list()`
Destruktor usuwa wszystkie elementy ze stosu za pomocą funkcji `pop`.
- `void test (unsigned long int length)`
Metoda `test()` realizuje operacje wypełniania listy danymi.

Metody prywatne

- `void push (int insert, unsigned int where)`
Metoda `push()` dodaje element na listę
- `void pop (unsigned int whence)`
Metoda `pop()` definiuje usuwanie elementu z listy.
- `unsigned size ()`
Metoda `size()` zwraca ilość elementów znajdujących się na liście.

Atrybuty prywatne

- `node * head`
Pole będące wskaźnikiem na pierwszy element listy.
- `node * tail`
Pole będące wskaźnikiem na ostatni element listy.

4.2.1 Opis szczegółowy

Definicja w linii 46 pliku `list.hh`.

4.2.2 Dokumentacja konstruktora i destruktora

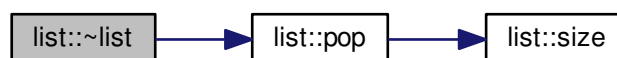
4.2.2.1 `list::list ()`

Definicja w linii 9 pliku `list.cpp`.

4.2.2.2 `list::~list ()`

Definicja w linii 15 pliku `list.cpp`.

Oto graf wywołań dla tej funkcji:



4.2.3 Dokumentacja funkcji składowych

4.2.3.1 void list::pop (unsigned int *whence*) [private]

Metoda `pop()` usuwa z listy wybrany element , lub zwraca błąd jeżeli lista jest już pusta.

Parametry

<i>in</i>	<i>whence</i>	- numer usuwanego elementu
-----------	---------------	----------------------------

Definicja w linii 73 pliku list.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.2.3.2 void list::push (int *insert*, unsigned int *where*) [private]

Metoda [push\(\)](#) wczytuje liczbę naturalną na listę.

Parametry

<i>in</i>	<i>insert</i>	- wartość dodawanego elementu
<i>in</i>	<i>where</i>	- na które miejsce ów element ma zostać dodany

Elementy listy są numerowane od 0!!

Przykład użycia funkcji:

push(3,3) -> wstawia element o wartości 3 na 3 miejsce listy , lub zwraca błąd 3 jeżeli lista jest zbyt krótka

Definicja w linii 29 pliku list.cpp.

Oto graf wywołań dla tej funkcji:



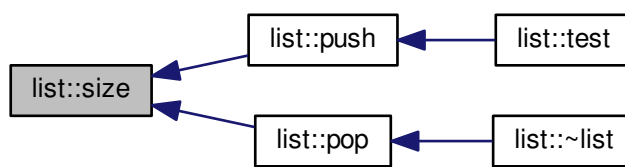
Oto graf wywoływań tej funkcji:



4.2.3.3 unsigned list::size () [private]

Definicja w linii 125 pliku list.cpp.

Oto graf wywoływań tej funkcji:



4.2.3.4 void list::test (unsigned long int length) [virtual]

Metoda `test()` realizuje wczytywanie zadanej ilości danych.

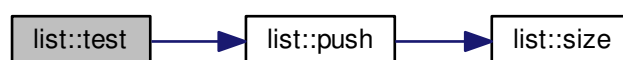
Parametry

in	<i>length</i>	- ilość dodawanych lementów
----	---------------	-----------------------------

Implementuje `benchmark`.

Definicja w linii 143 pliku list.cpp.

Oto graf wywołań dla tej funkcji:



4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 `node* list::head` [private]

Definicja w linii 55 pliku list.hh.

4.2.4.2 `node* list::tail` [private]

Definicja w linii 61 pliku list.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [list.hh](#)
- [list.cpp](#)

4.3 Dokumentacja klasy list_array

```
#include <list_array.hh>
```

Diagram dziedziczenia dla list_array

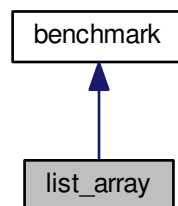
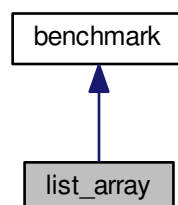


Diagram współpracy dla list_array:



Metody publiczne

- void [push](#) (int insert, unsigned int where, unsigned int extend, char x)

Metoda `push()` dodaje element na listę

- void `pop` (unsigned int whence)

Metoda `pop()` definiuje usuwanie elementu z listy.

- unsigned `size` ()

Metoda `size()` zwraca ilość elementów na liście.

- `list_array` ()

Konstruktor inicjalizuje pustą listę , wskaźnik na listę jest domyślnie ustawiony na NULL.

- `~list_array` ()

Destruktor usuwa listę

- void `test` (unsigned long int length)

Metoda `test()` realizuje operacje wypełniania listy danymi.

Atrybuty publiczne

- unsigned long int `n`

Ilość elementów na liście.

- int * `tmp`

Uchwyt do listy.

- unsigned long int `temp`

Rozmiar tablicy.

4.3.1 Opis szczegółowy

Definicja w linii 15 pliku `list_array.hh`.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 `list_array::list_array ()`

Definicja w linii 8 pliku `list_array.cpp`.

4.3.2.2 `list_array::~~list_array ()`

Definicja w linii 16 pliku `list_array.cpp`.

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 void `list_array::pop (unsigned int whence)`

Metoda `pop()` usuwa z listy ostatni/pierwszy element lub zwraca komunikat o błędzie w przypadku próby usunięcia elementu z pustej listy.

Parametry

<code>in</code>	<code>whence</code>	- numer usuwanego elementu
-----------------	---------------------	----------------------------

Definicja w linii 160 pliku `list_array.cpp`.

Oto graf wywołań dla tej funkcji:



4.3.3.2 void list_array::push (int *insert*, unsigned int *where*, unsigned int *extend*, char *x*)

Metoda `push()` wczytuje liczbę naturalną na początek, koniec lub w wybrane miejsce listy;

Parametr *x* może przyjmować 2 wartości '*' lub '+' co odpowiada powiększeniu tablicy o *x* lub *x* razy

Przykład wywołania funkcji :

`push(10,2,4,'*')` - Na drugie miejsce w liście zostanie wstawiona liczba 10 , w przypadku przekroczenia rozmiaru listy jej rozmiar zostanie zwiększony 4 razy.

Parametry

in	<i>insert</i>	- wartość dodawanego elementu
in	<i>where</i>	- na które miejsce ów element ma zostać dodany
in	<i>extend</i>	- parametr ustalający o ile powiększyć listę
in	<i>x</i>	- typ zwiększania tablicy '+'/'-'

Definicja w linii 29 pliku `list_array.cpp`.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

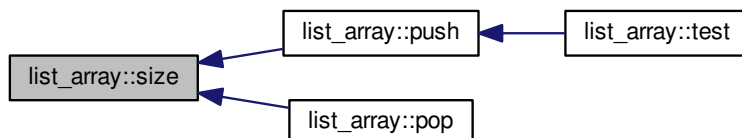


4.3.3.3 `unsigned list_array::size ()`

Metoda `size()` zwraca ilość elementów znajdujących się na liście.

Definicja w linii 199 pliku `list_array.cpp`.

Oto graf wywoływań tej funkcji:

4.3.3.4 `void list_array::test (unsigned long int length) [virtual]`

Metoda `test()` realizuje wczytywanie zadanej ilości danych do listy.

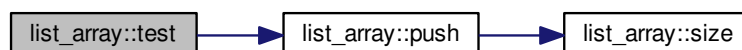
Parametry

<code>in</code>	<code>length</code>	- ilość dodawanych lementów
-----------------	---------------------	-----------------------------

Implementuje `benchmark`.

Definicja w linii 207 pliku `list_array.cpp`.

Oto graf wywołań dla tej funkcji:



4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 `unsigned long int list_array::n`

Definicja w linii 22 pliku `list_array.hh`.

4.3.4.2 `unsigned long int list_array::temp`

Definicja w linii 30 pliku `list_array.hh`.

4.3.4.3 `int* list_array::tmp`

Definicja w linii 26 pliku `list_array.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [list_array.hh](#)
- [list_array.cpp](#)

4.4 Dokumentacja struktury node

```
#include <list.hh>
```

Diagram współpracy dla node:



Metody publiczne

- [node](#) (int element)
Konstruktor węzła.

Atrybuty publiczne

- int [data](#)
Pole do którego dopisywane są dane.
- [node](#) * [next](#)
Pole będące wskaźnikiem na następny element.

4.4.1 Opis szczegółowy

Definicja w linii 16 pliku list.hh.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 node::node (int *element*) [inline]

Konstruktor inicjalizuje nowy węzeł z wartością równą element oraz wskaźnikiem na NULL

Parametry

in	<i>element</i>	- element dodawany na koniec listy
----	----------------	------------------------------------

Definicja w linii 38 pliku list.hh.

4.4.3 Dokumentacja atrybutów składowych

4.4.3.1 int node::data

Definicja w linii 22 pliku list.hh.

4.4.3.2 node* node::next

Definicja w linii 28 pliku list.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [list.hh](#)

4.5 Dokumentacja klasy queue

```
#include <queue.hh>
```

Diagram dziedziczenia dla queue

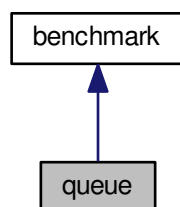
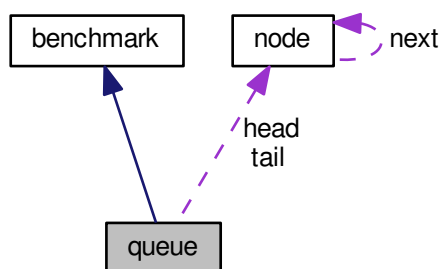


Diagram współpracy dla queue:



Metody publiczne

- [queue\(\)](#)
Konstruktor inicjalizujący zmienną wskaźnikową, która domyślnie ma pokazywać na NULL.
- [~queue\(\)](#)
Destruktor usuwa wszystkie elementy z kolejki za pomocą funkcji pop.
- void [test](#) (unsigned long int length)
Metoda [test\(\)](#) realizuje operacje zapelniania kolejki ustalonymi danymi, czas będzie zliczany.

Metody prywatne

- void `push` (int insert)
Metoda `push()` dodaje daną do kolejki.
- void `pop` ()
Metoda `pop()` definiuje usuwanie elementu z kolejki.
- unsigned `size` ()
Metoda `size()` zwraca ilość elementów kolejki.

Atrybuty prywatne

- `node * head`
Pole będące pierwszym wskaźnikiem na elementy kolejki.
- `node * tail`
Pole będące wskaźnikiem na ostatni element kolejki.

4.5.1 Opis szczegółowy

Definicja w linii 18 pliku `queue.hh`.

4.5.2 Dokumentacja konstruktora i destruktor

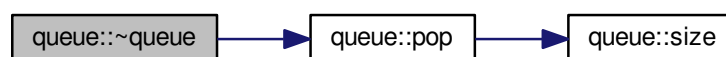
4.5.2.1 `queue::queue ()`

Definicja w linii 9 pliku `queue.cpp`.

4.5.2.2 `queue::~~queue ()`

Definicja w linii 14 pliku `queue.cpp`.

Oto graf wywołań dla tej funkcji:



4.5.3 Dokumentacja funkcji składowych

4.5.3.1 `void queue::pop () [private]`

Metoda `pop()` usuwa z kolejki pierwszy element lub zwraca komunikat o błędzie w przypadku próby usunięcia elementu z pustej kolejki.

Definicja w linii 47 pliku `queue.cpp`.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.2 void queue::push (int *insert*) [private]

Metoda `push()` wczytuje liczbę naturalną do kolejki

Przykład wywołania funkcji :

`push(10)` - Na koniec kolejki zostanie wprowadzona liczba 10.

Parametry

<code>in</code>	<code>insert</code>	- dodawany element
-----------------	---------------------	--------------------

Definicja w linii 27 pliku `queue.cpp`.

Oto graf wywoływań tej funkcji:

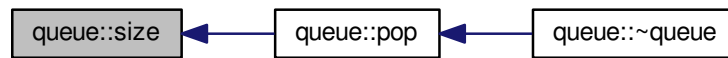


4.5.3.3 unsigned queue::size () [private]

Metoda `size()` zwraca ilość elementów znajdujących się w kolejce.

Definicja w linii 68 pliku `queue.cpp`.

Oto graf wywoływań tej funkcji:



4.5.3.4 void queue::test (unsigned long int *length*) [virtual]

Metoda `test()` realizuje wczytywanie zadanej ilości danych do kolejki.

Parametry

<code>in</code>	<code>length</code>	- ilość danych do wstawienia
-----------------	---------------------	------------------------------

Implementuje [benchmark](#).

Definicja w linii 83 pliku `queue.cpp`.

Oto graf wywołań dla tej funkcji:



4.5.4 Dokumentacja atrybutów składowych

4.5.4.1 node* queue::head [private]

Definicja w linii 27 pliku `queue.hh`.

4.5.4.2 node* queue::tail [private]

Definicja w linii 31 pliku `queue.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [queue.hh](#)
- [queue.cpp](#)

4.6 Dokumentacja klasy stack

```
#include <stack.hh>
```

Diagram dziedziczenia dla stack

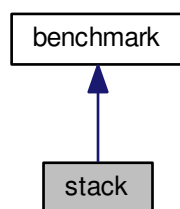
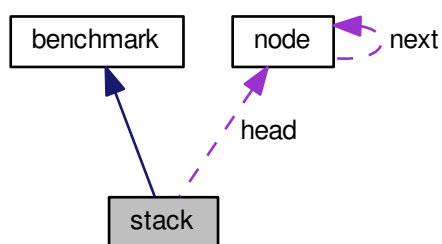


Diagram współpracy dla stack:



Metody publiczne

- `stack()`
Konstruktor inicjalizujący pusty stos, początek stosu jest domyślnie ustawiony na NULL.
- `~stack()`
Destruktor usuwa wszystkie elementy ze stosu za pomocą funkcji pop.
- `void test(unsigned long int length)`
Metoda `test()` realizuje operacje zapelniania stosu ustalonymi danymi, czas będzie zliczany.

Metody prywatne

- `void push(int insert)`
Metoda `push()` dodaje daną na stos.
- `void pop()`
Metoda `pop()` definiuje usuwanie elementu ze stosu.
- `unsigned size()`
Metoda `size()` zwraca ilość elementów stosu.

Atrybuty prywatne

- `node * head`

Pole będące wskaźnikiem na pierwszy element stosu.

4.6.1 Opis szczegółowy

Definicja w linii 15 pliku `stack.hh`.

4.6.2 Dokumentacja konstruktora i destruktor

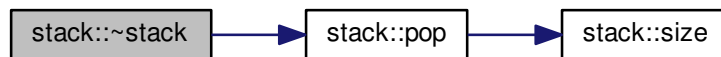
4.6.2.1 `stack::stack ()`

Definicja w linii 8 pliku `stack.cpp`.

4.6.2.2 `stack::~~stack ()`

Definicja w linii 12 pliku `stack.cpp`.

Oto graf wywołań dla tej funkcji:



4.6.3 Dokumentacja funkcji składowych

4.6.3.1 `void stack::pop () [private]`

Metoda `pop()` usuwa ze stosu ostatni element lub zwraca komunikat o błędzie w przypadku próby usunięcia elementu z pustego stosu.

Definicja w linii 39 pliku `stack.cpp`.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.6.3.2 `void stack::push (int insert) [private]`

Metoda `push()` wczytuje liczbę naturalną na stos

Przykład wywołania funkcji :

`push(10)` - Na początek stosu zostanie wprowadzona liczba 10.

Parametry

<code>in</code>	<code>insert</code>	- wartość dodawanego elementu
-----------------	---------------------	-------------------------------

Definicja w linii 22 pliku `stack.cpp`.

Oto graf wywoływań tej funkcji:

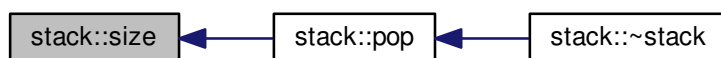


4.6.3.3 `unsigned stack::size () [private]`

Metoda `size()` zwraca ilość elementów znajdujących się na stosie.

Definicja w linii 56 pliku `stack.cpp`.

Oto graf wywoływań tej funkcji:



4.6.3.4 `void stack::test (unsigned long int length) [virtual]`

Metoda `test()` realizuje wczytywanie zadanej ilości danych na stos.

Parametry

<code>in</code>	<code>length</code>	- ilość dodawanych elementów
-----------------	---------------------	------------------------------

Implementuje [benchmark](#).

Definicja w linii 71 pliku `stack.cpp`.

Oto graf wywołań dla tej funkcji:



4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 `node* stack::head` `[private]`

Definicja w linii 24 pliku `stack.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [stack.hh](#)
- [stack.cpp](#)

Rozdział 5

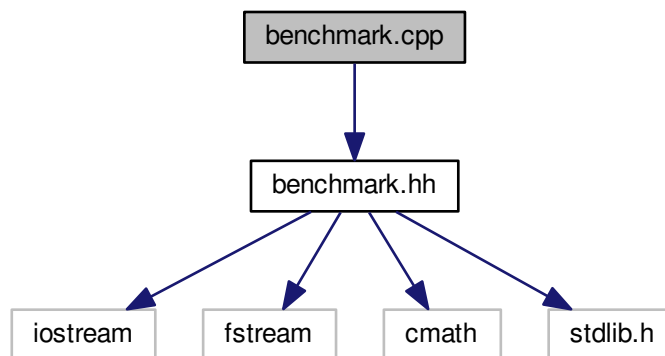
Dokumentacja plików

5.1 Dokumentacja pliku benchmark.cpp

Deklaracja funkcji z klasy Benchmark.

```
#include "benchmark.hh"
```

Wykres zależności załączania dla benchmark.cpp:

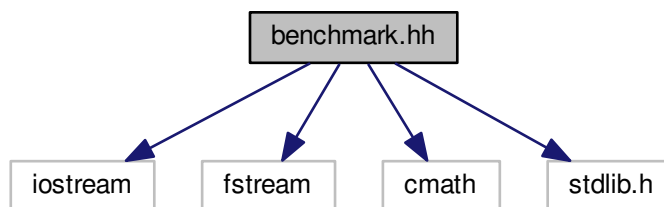


5.2 Dokumentacja pliku benchmark.hh

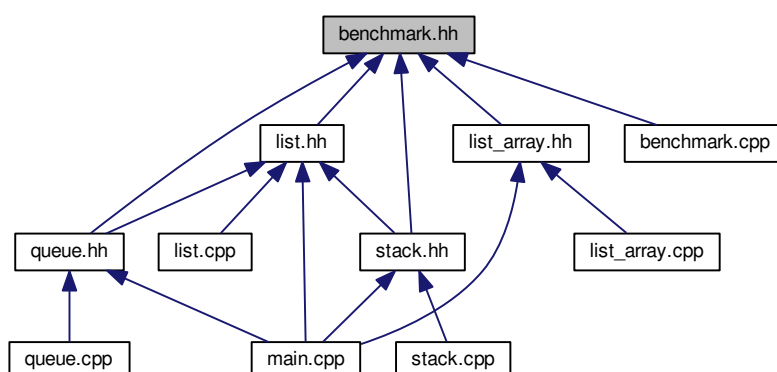
Definicja klasy Benchmark.

```
#include <iostream>
#include <fstream>
#include <cmath>
#include "stdlib.h"
```

Wykres zależności załączania dla `benchmark.hh`:



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



Komponenty

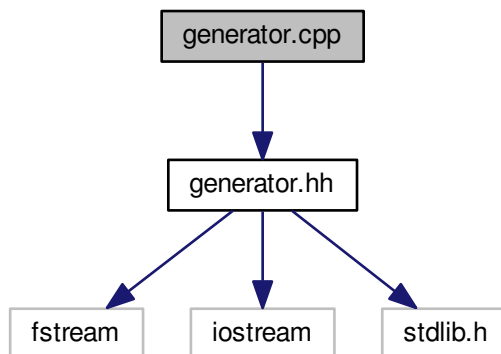
- class `benchmark`

5.3 Dokumentacja pliku `generator.cpp`

Deklaracja funkcji generującej liczby losowe.

```
#include "generator.hh"
```

Wykres zależności załączania dla generator.cpp:



Funkcje

- void [data_generator](#) (unsigned long int data_amount)

Generuje liczby losowe.

5.3.1 Dokumentacja funkcji

5.3.1.1 void data_generator (unsigned long int data_amount)

Funkcja generuje naturalne liczby losowe z przedziału 0-100, które następnie są zapisywane do pliku random_data.dat

Parametry

in	data_amount	- ilość liczb wynikowych które chcemy uzyskać
----	-------------	---

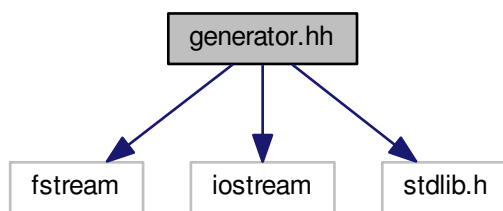
Definicja w linii 9 pliku generator.cpp.

5.4 Dokumentacja pliku generator.hh

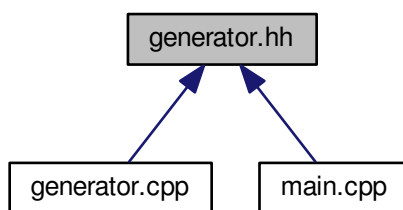
Definicja generatora liczb losowych.

```
#include <fstream>
#include <iostream>
#include <stdlib.h>
```

Wykres zależności załączania dla generator.hh:



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



Funkcje

- void [data_generator](#) (unsigned long int data_amount)
Generuje liczby losowe.

5.4.1 Dokumentacja funkcji

5.4.1.1 void data_generator (unsigned long int data_amount)

Funkcja generuje naturalne liczby losowe z przedziału 0-100, które następnie są zapisywane do pliku random_data.dat

Parametry

in	data_amount	- ilość liczb wynikowych które chcemy uzyskać
----	-------------	---

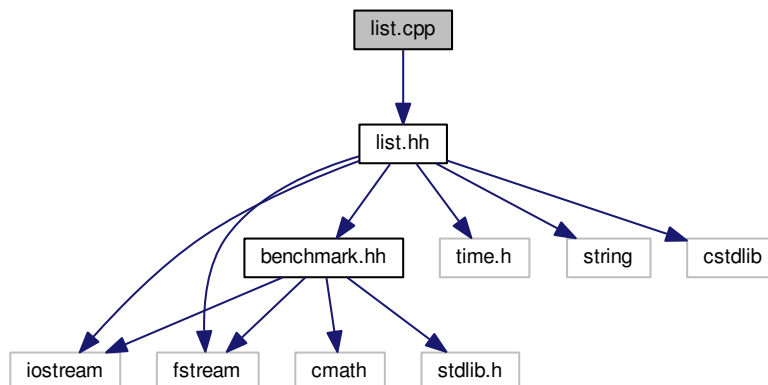
Definicja w linii 9 pliku generator.cpp.

5.5 Dokumentacja pliku list.cpp

Deklaracja klasy list.

```
#include "list.hh"
```

Wykres zależności załączania dla list.cpp:

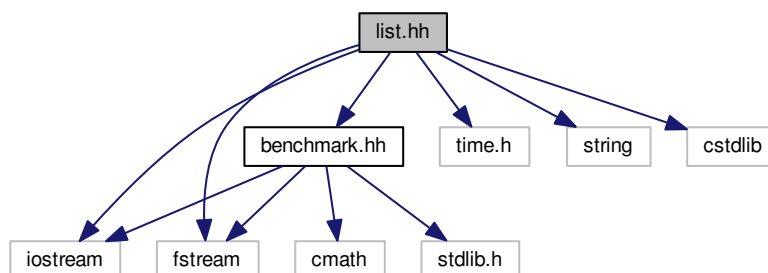


5.6 Dokumentacja pliku list.hh

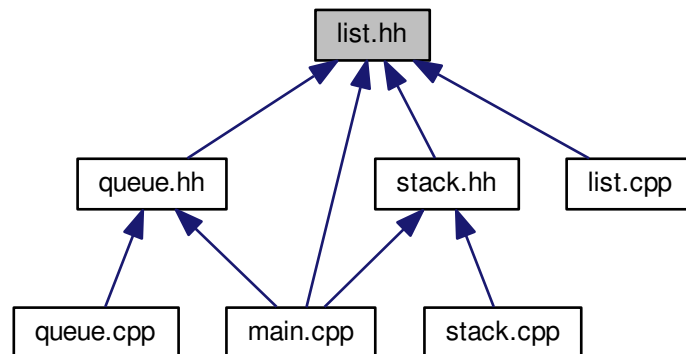
Definicja klasy lista.

```
#include <iostream>
#include <time.h>
#include <string>
#include <fstream>
#include <cstdlib>
#include "benchmark.hh"
```

Wykres zależności załączania dla list.hh:



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



Komponenty

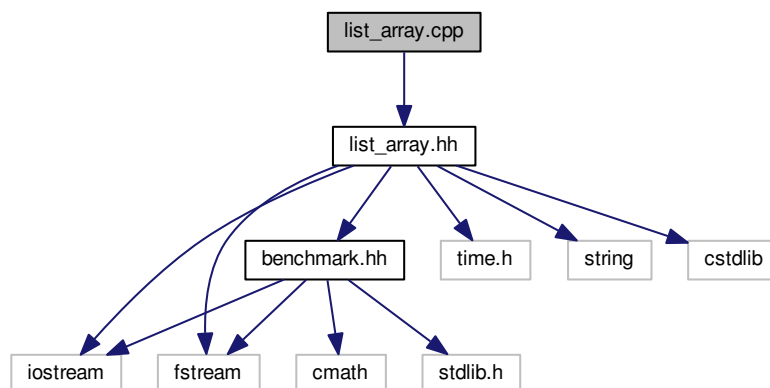
- struct [node](#)
- class [list](#)

5.7 Dokumentacja pliku list_array.cpp

Deklaracja klasy list.

```
#include "list_array.hh"
```

Wykres zależności załączania dla list_array.cpp:

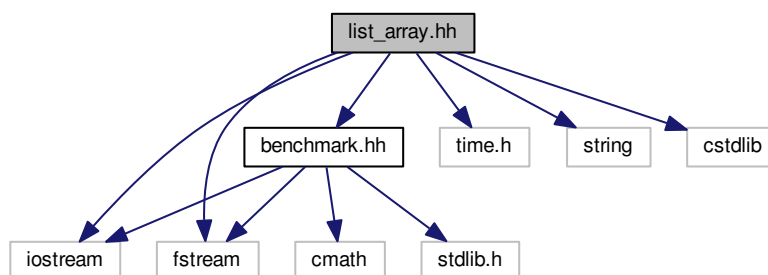


5.8 Dokumentacja pliku list_array.hh

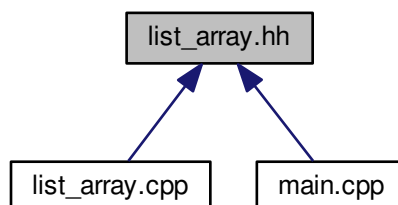
Definicja klasy `list_array`.

```
#include <iostream>
#include <time.h>
#include <string>
#include <fstream>
#include <cstdlib>
#include "benchmark.hh"
```

Wykres zależności załączania dla list_array.hh:



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



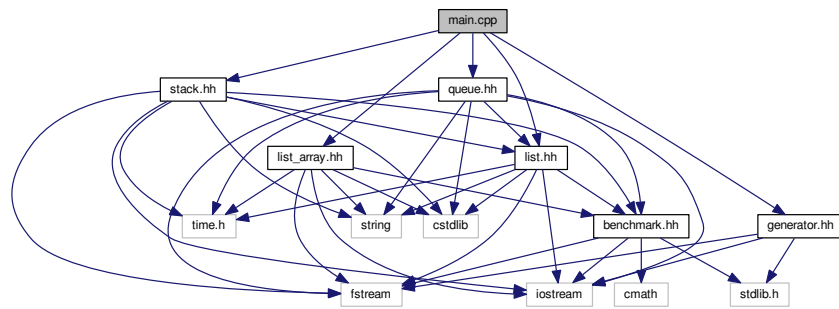
Komponenty

- class `list_array`

5.9 Dokumentacja pliku main.cpp

```
#include "stack.hh"
#include "generator.hh"
#include "list.hh"
#include "queue.hh"
#include "list_array.hh"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int `main()`

Aby wygenerować liczby losowe należy odkomentować linie zawierającą funkcję `data_generator()`

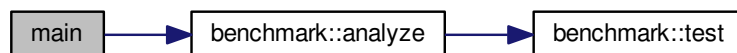
5.9.1 Dokumentacja funkcji

5.9.1.1 int main ()

Generuj Dane losowe (ustawione na 10^7)

Definicja w linii 9 pliku main.cpp.

Oto graf wywołań dla tej funkcji:

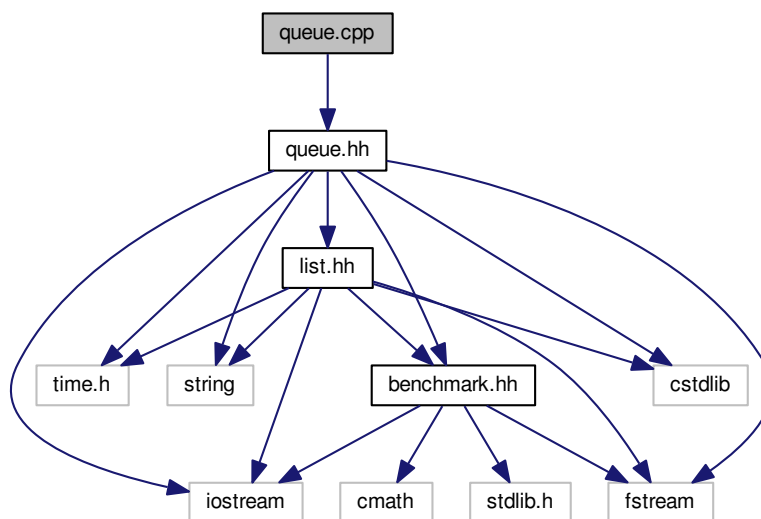


5.10 Dokumentacja pliku queue.cpp

Deklaracja klasy queue.

```
#include "queue.hh"
```

Wykres zależności załączania dla queue.cpp:



5.11 Dokumentacja pliku queue.hh

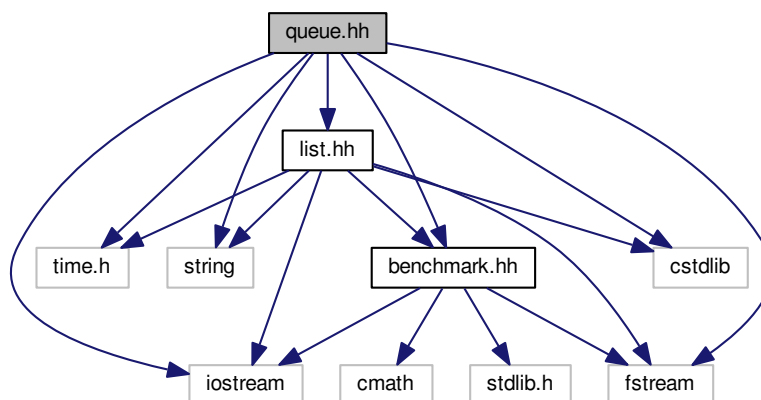
Definicja klasy stack.

```

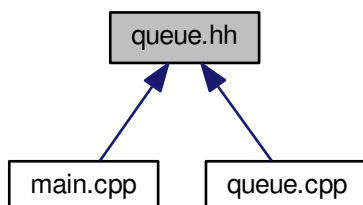
#include <iostream>
#include <time.h>
#include <string>
#include <fstream>
#include <cstdlib>
#include "benchmark.hh"
#include "list.hh"

```

Wykres zależności załączania dla queue.hh:



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



Komponenty

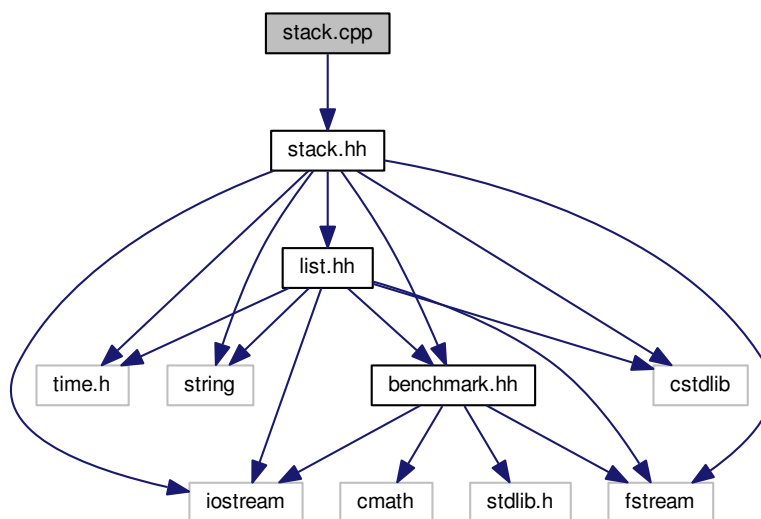
- class `queue`

5.12 Dokumentacja pliku stack.cpp

Deklaracja klasy stack.

```
#include "stack.hh"
```

Wykres zależności załączania dla stack.cpp:

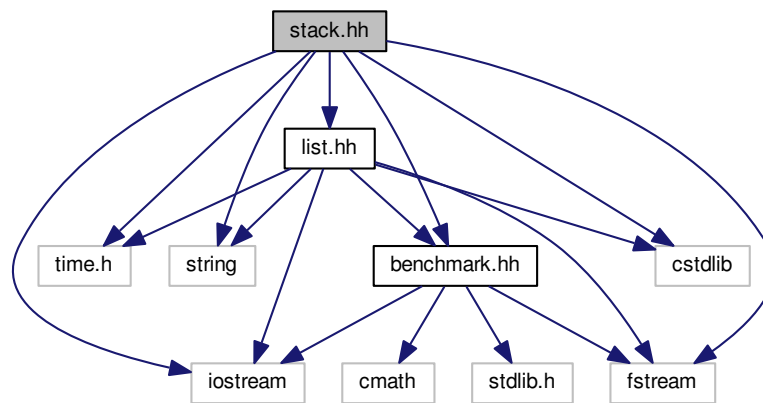


5.13 Dokumentacja pliku stack.hh

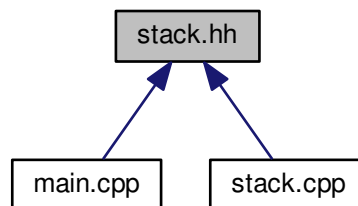
Definicja klasy stack.

```
#include <iostream>
#include <time.h>
#include <string>
#include <fstream>
#include <cstdlib>
#include "benchmark.hh"
#include "list.hh"
```

Wykres zależności załączania dla `stack.hh`:



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



Komponenty

- class [stack](#)

Skorowidz

- ~list
 - list, [10](#)
- ~list_array
 - list_array, [15](#)
- ~queue
 - queue, [20](#)
- ~stack
 - stack, [24](#)
- analyze
 - benchmark, [7](#)
- benchmark, [7](#)
 - analyze, [7](#)
 - test, [8](#)
- benchmark.cpp, [29](#)
- benchmark.hh, [29](#)
- data
 - node, [18](#)
- data_generator
 - generator.cpp, [31](#)
 - generator.hh, [32](#)
- generator.cpp, [30](#)
 - data_generator, [31](#)
- generator.hh, [31](#)
 - data_generator, [32](#)
- head
 - list, [14](#)
 - queue, [22](#)
 - stack, [27](#)
- list, [9](#)
 - ~list, [10](#)
 - head, [14](#)
 - list, [10](#)
 - pop, [11](#)
 - push, [12](#)
 - size, [13](#)
 - tail, [14](#)
 - test, [13](#)
- list.cpp, [32](#)
- list.hh, [33](#)
- list_array, [14](#)
 - ~list_array, [15](#)
 - list_array, [15](#)
 - list_array, [15](#)
 - n, [17](#)
 - pop, [15](#)
 - push, [16](#)
 - size, [16](#)
 - temp, [17](#)
 - test, [17](#)
 - tmp, [17](#)
- list_array.cpp, [34](#)
- list_array.hh, [35](#)
- main
 - main.cpp, [36](#)
 - main.cpp, [35](#)
 - main, [36](#)
- n
 - list_array, [17](#)
- next
 - node, [18](#)
- node, [18](#)
 - data, [18](#)
 - next, [18](#)
 - node, [18](#)
- pop
 - list, [11](#)
 - list_array, [15](#)
 - queue, [20](#)
 - stack, [24](#)
- push
 - list, [12](#)
 - list_array, [16](#)
 - queue, [21](#)
 - stack, [25](#)
- queue, [19](#)
 - ~queue, [20](#)
 - head, [22](#)
 - pop, [20](#)
 - push, [21](#)
 - queue, [20](#)
 - size, [21](#)
 - tail, [22](#)
 - test, [22](#)
- queue.cpp, [36](#)
- queue.hh, [37](#)
- size
 - list, [13](#)
 - list_array, [16](#)
 - queue, [21](#)
 - stack, [25](#)

- stack, [22](#)
 - ~stack, [24](#)
 - head, [27](#)
 - pop, [24](#)
 - push, [25](#)
 - size, [25](#)
 - stack, [24](#)
 - test, [25](#)
- stack.cpp, [38](#)
- stack.hh, [39](#)
- tail
 - list, [14](#)
 - queue, [22](#)
- temp
 - list_array, [17](#)
- test
 - benchmark, [8](#)
 - list, [13](#)
 - list_array, [17](#)
 - queue, [22](#)
 - stack, [25](#)
- tmp
 - list_array, [17](#)