Benchmark + Mnozenie

0.3

Wygenerowano przez Doxygen 1.8.6

Cz, 19 mar 2015 11:22:35

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 rozpocznij_pomiar . . . 7

4.1.2.2 testuj . . . 8

4.1.2.3 testuj_strukture . . . 8

4.1.2.4 zakoncz_pomiar . . . 9

4.1.3 Dokumentacja atrybutów składowych . . . 10

4.1.3.1 czas_pomiaru . . . 10

4.1.3.2 t1 . . . 10

4.1.3.3 t2 . . . 10

4.2 Dokumentacja klasy Kolejka . . . 10

4.2.1 Opis szczegółowy . . . 11

4.2.2 Dokumentacja konstruktora i destruktora . . . 11

4.2.2.1 Kolejka . . . 11

4.2.3 Dokumentacja funkcji składowych . . . 12

4.2.3.1 pop . . . 12

4.2.3.2 push . . . 12

4.2.3.3 size . . . 12

4.2.3.4 wyczysc_dane . . . 12

4.2.3.5 wykonaj_program . . . 13

4.2.4 Dokumentacja atrybutów składowych . . . 13

4.2.4.1 head . . . 13
4.2.4.2 tail . . . 13
4.3 Dokumentacja klasy Lista . . . 14
4.3.1 Opis szczegółowy . . . 15
4.3.2 Dokumentacja konstruktora i destruktora . . . 15
4.3.2.1 Lista . . . 15
4.3.3 Dokumentacja funkcji składowych . . . 15
4.3.3.1 pop . . . 15
4.3.3.2 push . . . 15
4.3.3.3 size . . . 16
4.3.3.4 wyczysc_dane . . . 16
4.3.3.5 wykonaj_program . . . 16
4.3.4 Dokumentacja atrybutów składowych . . . 17
4.3.4.1 first . . . 17
4.4 Dokumentacja klasy Lista_tab . . . 17
4.4.1 Opis szczegółowy . . . 18
4.4.2 Dokumentacja konstruktora i destruktora . . . 19
4.4.2.1 Lista_tab . . . 19
4.4.2.2 ∼Lista_tab . . . 19
4.4.3 Dokumentacja funkcji składowych . . . 19
4.4.3.1 pop . . . 19
4.4.3.2 push . . . 19
4.4.3.3 size . . . 19
4.4.3.4 wyczysc_dane . . . 20
4.4.3.5 wykonaj_program . . . 21
4.4.4 Dokumentacja atrybutów składowych . . . 21
4.4.4.1 iterator . . . 21
4.4.4.2 rozmiar . . . 21
4.4.4.3 tab . . . 21
4.5 Dokumentacja struktury pole . . . 21
4.5.1 Opis szczegółowy . . . 22
4.5.2 Dokumentacja konstruktora i destruktora . . . 22
4.5.2.1 pole . . . 22
4.5.3 Dokumentacja atrybutów składowych . . . 22
4.5.3.1 next . . . 22
4.5.3.2 wartosc . . . 22
4.6 Dokumentacja klasy Program . . . 22
4.6.1 Opis szczegółowy . . . 23
4.6.2 Dokumentacja konstruktora i destruktora . . . 24
4.6.2.1 Program . . . 24

4.6.2.2 ~Program . . . 24
4.6.3 Dokumentacja funkcji składowych . . . 24
4.6.3.1 getRozmiar_tab . . . 24
4.6.3.2 wczytaj_dane . . . 24
4.6.3.3 wczytaj_dane . . . 24
4.6.3.4 wyczysc_dane . . . 25
4.6.3.5 wykonaj_program . . . 25
4.6.3.6 wykonaj_program . . . 25
4.6.3.7 wyswietl_dane . . . 25
4.6.3.8 zapisz_dane . . . 26
4.6.4 Dokumentacja atrybutów składowych . . . 26
4.6.4.1 plik_we . . . 26
4.6.4.2 plik_wy . . . 26
4.6.4.3 rozmiar_tab . . . 26
4.6.4.4 tab . . . 26
4.7 Dokumentacja klasy Stos . . . 26
4.7.1 Opis szczegółowy . . . 28
4.7.2 Dokumentacja konstruktora i destruktora . . . 28
4.7.2.1 Stos . . . 28
4.7.3 Dokumentacja funkcji składowych . . . 28
4.7.3.1 pop . . . 28
4.7.3.2 push . . . 28
4.7.3.3 size . . . 29
4.7.3.4 wyczysc_dane . . . 29
4.7.3.5 wykonaj_program . . . 29
4.7.4 Dokumentacja atrybutów składowych . . . 30
4.7.4.1 top . . . 30
4.8 Dokumentacja klasy Tabx2 . . . 30
4.8.1 Opis szczegółowy . . . 31
4.8.2 Dokumentacja funkcji składowych . . . 31
4.8.2.1 wykonaj_program . . . 31

5 Dokumentacja plików **33**

5.1 Dokumentacja pliku benchmark.cpp . . . 33
5.2 benchmark.cpp . . . 34
5.3 Dokumentacja pliku benchmark.hh . . . 34
5.4 benchmark.hh . . . 35
5.5 Dokumentacja pliku kolejka.cpp . . . 36
5.6 kolejka.cpp . . . 37
5.7 Dokumentacja pliku kolejka.hh . . . 37

5.8 kolejka.hh . . . 38
5.9 Dokumentacja pliku lista.cpp . . . 39
5.10 lista.cpp . . . 39
5.11 Dokumentacja pliku lista.hh . . . 40
5.12 lista.hh . . . 41
5.13 Dokumentacja pliku lista_tab.cpp . . . 42
5.14 lista_tab.cpp . . . 42
5.15 Dokumentacja pliku lista_tab.hh . . . 44
5.15.1 Dokumentacja definicji . . . 44
5.15.1.1 LISTA__TAB_HH . . . 45
5.16 lista_tab.hh . . . 45
5.17 Dokumentacja pliku main.cpp . . . 45
5.17.1 Dokumentacja funkcji . . . 46
5.17.1.1 main . . . 46
5.18 main.cpp . . . 46
5.19 Dokumentacja pliku program.cpp . . . 47
5.20 program.cpp . . . 47
5.21 Dokumentacja pliku program.hh . . . 48
5.22 program.hh . . . 49
5.23 Dokumentacja pliku stos.cpp . . . 50
5.24 stos.cpp . . . 50
5.25 Dokumentacja pliku stos.hh . . . 51
5.26 stos.hh . . . 52
5.27 Dokumentacja pliku tabx2.cpp . . . 53
5.28 tabx2.cpp . . . 53
5.29 Dokumentacja pliku tabx2.hh . . . 54
5.29.1 Opis szczegółowy . . . 54
5.30 tabx2.hh . . . 55

Indeks **56**

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

- Benchmark . . . 7
- pole . . . 21
- Program . . . 22
  - Kolejka . . . 10
  - Lista . . . 14
  - Lista_tab . . . 17
  - Stos . . . 26
  - Tabx2 . . . 30

Rozdział 2

Indeks klas

2.1 Lista klas

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

Benchmark
: Klasa Benchmark 7
Kolejka 10
Lista 14
Lista_tab 17
pole
: Struktura pole 21
Program
: Modeluje klase Program 22
Stos 26
Tabx2 30

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

benchmark.cpp
Plik zawiera metody klasy Benchmark . . . 34
benchmark.hh
Definicja klasy Benchmark . . . 35
kolejka.cpp
Zawiera definicje metod klasy Kolejka . . . 37
kolejka.hh
Definicja klasy Kolejka . . . 38
lista.cpp
Zawiera definicje metod klasy Lista . . . 39
lista.hh
Definicja klasy Lista . . . 41
lista_tab.cpp
Zawiera definicje metod klasy Lista . . . 42
lista_tab.hh
Definicja klasy Lista_tab . . . 45
main.cpp . . . 46
program.cpp
Plik zawiera metody klasy Program . . . 47
program.hh
Definicja klasy Program . . . 49
stos.cpp
Zawiera definicje metod klasy Stos . . . 50
stos.hh
Definicja klasy Stos . . . 52
tabx2.cpp
Plik zawiera metody klasy Tabx2 . . . 53
tabx2.hh
Definicja klasy Tabx2 . . . 55

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy Benchmark

Klasa Benchmark.

`#include <benchmark.hh>`

Metody publiczne

- void rozpocznij_pomiar ()

  Procedura rozpocznij_pomiar.
- void zakoncz_pomiar ()

  Procedura zakoncz_pomiar.
- double testuj (Program &program, char *dane, int ilosc_danych, int ilosc_testow)

  Metoda testuj.
- double testuj_strukture (Program &program, char *dane, int ilosc_danych, int ilosc_testow)

  Metoda testuj_strukture.

Atrybuty prywatne

- timeval t1

  Zmienne t1, t2.
- timeval t2
- double czas_pomiaru

  Zmienna czas_pomiaru.

4.1.1 Opis szczegółowy

Jest to klasa służąca do testowania programów.

Definicja w linii 23 pliku benchmark.hh.

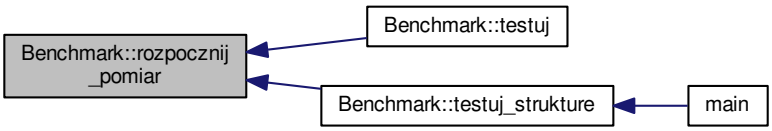
4.1.2 Dokumentacja funkcji składowych

4.1.2.1 void Benchmark::rozpocznij_pomiar ( )

Rozpoczyna pomiar czasu.

Definicja w linii 7 pliku benchmark.cpp.

Oto graf wywoływań tej funkcji:



4.1.2.2 double Benchmark::testuj (Program & *program*, char * *dane*, int *ilosc_danych*, int *ilosc_testow*)

Dokonuje testow wybranego programu. Dane wczytywane sa do tablicy.

Parametry

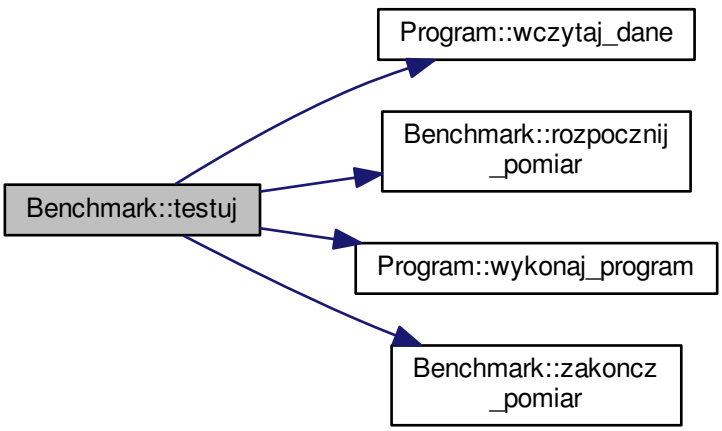
in	*program*	Program wybrany do testowania.
in	*dane*	Wskaznik na nazwe pliku z danymi.
in	*ilosc_danych*	Ilosc danych, ktore chcemy pobrac do testu.
in	*ilosc_testow*	Ilosc testow, jakie chcemy przeprowadzic.

Zwraca

Metoda zwraca sredni czas wykonania programu dla podanych parametrow.

Definicja w linii 17 pliku benchmark.cpp.

Oto graf wywołań dla tej funkcji:



4.1.2.3 double Benchmark::testuj_strukture (Program & *program*, char * *dane*, int *ilosc_danych*, int *ilosc_testow*)

Dokonuje testow wybranego programu. Dane wczytywane sa do odpowiedniej struktury.

Parametry

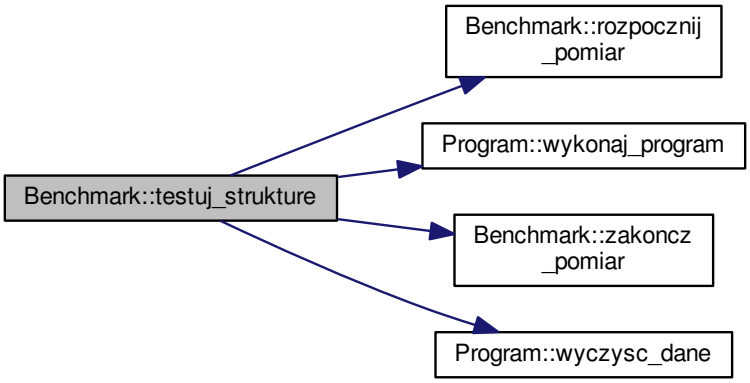
in	*program*	Program wybrany do testowania.
in	*dane*	Wskaznik na nazwe pliku z danymi.
in	*ilosc_danych*	Ilosc danych, ktore chcemy pobrac do testu.
in	*ilosc_testow*	Ilosc testow, jakie chcemy przeprowadzic.

Zwraca

Metoda zwraca sredni czas wykonania programu dla podanych parametrow.

Definicja w linii 46 pliku benchmark.cpp.

Oto graf wywołań dla tej funkcji:



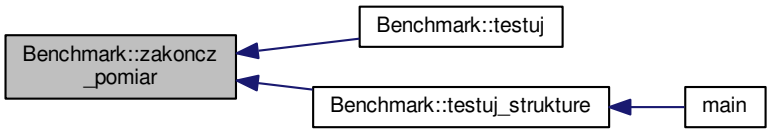
Oto graf wywoływań tej funkcji:

4.1.2.4 void Benchmark::zakoncz_pomiar ( )

Konczy pomiar czasu i zapisuje wartosc zmierzona w zmiennej czas_pomiaru.

Definicja w linii 11 pliku benchmark.cpp.

Oto graf wywoływań tej funkcji:



4.1.3 Dokumentacja atrybutów składowych

4.1.3.1 double Benchmark::czas_pomiaru [private]

Przechowuje obliczony czas pojedynczego pomiaru (w ms)

Definicja w linii 37 pliku benchmark.hh.

4.1.3.2 timeval Benchmark::t1 [private]

Zmienne przechowujace momenty rozpaczecia i zakonczenia pomiaru czasu.

Definicja w linii 30 pliku benchmark.hh.

4.1.3.3 timeval Benchmark::t2 [private]

Definicja w linii 30 pliku benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- benchmark.hh
- benchmark.cpp

4.2 Dokumentacja klasy Kolejka

`#include <kolejka.hh>`

Diagram dziedziczenia dla Kolejka

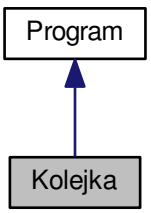
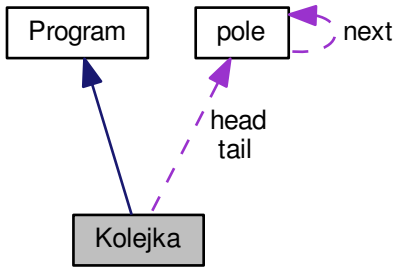


Diagram współpracy dla Kolejka:



Metody publiczne

- Kolejka ()
- void push (int x)

  Metoda push.
- void pop ()

  Procedura pop.
- int size ()

  Metoda size.
- bool wykonaj_program (char ∗nazwa_pliku, int ilosc_danych)

  Metoda wykonaj_program.
- void wyczysc_dane (int ile)

  Metoda wyczysc_dane.

Atrybuty prywatne

- pole ∗ head
- pole ∗ tail

Dodatkowe Dziedziczone Składowe

4.2.1 Opis szczegółowy

Definicja w linii 11 pliku kolejka.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Kolejka::Kolejka ( ) [inline]

\ brief Konstruktor bezparametryczny

Ustawia glowe (head) i ogon (tail) kolejki na NULL

Definicja w linii 21 pliku kolejka.hh.

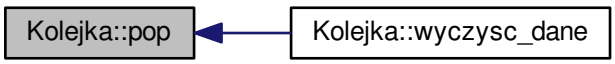
4.2.3 Dokumentacja funkcji składowych

4.2.3.1 void Kolejka::pop ( )

Usuwa element z poczatku kolejki.

Definicja w linii 20 pliku kolejka.cpp.

Oto graf wywoływań tej funkcji:



4.2.3.2 void Kolejka::push (int *x*)

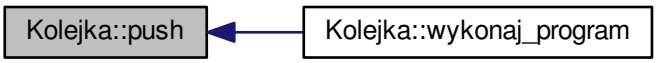
Dodaje podana wartosc na koniec kolejki.

Parametry

in	*x*	Wartosc, ktora chcemy dodac na koniec kolejki.

Definicja w linii 7 pliku kolejka.cpp.

Oto graf wywoływań tej funkcji:



4.2.3.3 int Kolejka::size ( )

Daje informacje o rozmiarze kolejki (liczbie jej elementow).

Zwraca

Rozmiar kolejki (liczba jej elementow)

Definicja w linii 31 pliku kolejka.cpp.

4.2.3.4 void Kolejka::wyczysc_dane (int *ile*) [virtual]

Usuwa zadana ilosc elementow kolejki za pomoca metody pop

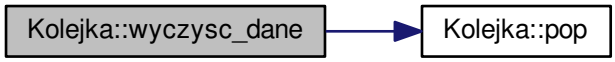
Parametry

in	*ile*	Liczba elementow, ktore chcemy usunac.

Implementuje Program.

Definicja w linii 61 pliku kolejka.cpp.

Oto graf wywołań dla tej funkcji:



4.2.3.5 bool Kolejka::wykonaj_program (char * *nazwa_pliku,* int *ilosc_danych*) [virtual]

Wykonuje zadany program - dodanie zadanej ilosci danych z pliku do kolejki za pomoca metody push()

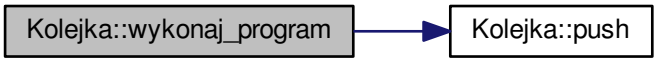
Zwracane wartości

TRUE	Poprawnie wykonano program
FALSE	Nie wczytano danych

Implementuje Program.

Definicja w linii 45 pliku kolejka.cpp.

Oto graf wywołań dla tej funkcji:



4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 pole* Kolejka::head [private]

Definicja w linii 12 pliku kolejka.hh.

4.2.4.2 pole* Kolejka::tail [private]

Definicja w linii 13 pliku kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- kolejka.hh
- kolejka.cpp

4.3 Dokumentacja klasy Lista

`#include <lista.hh>`

Diagram dziedziczenia dla Lista

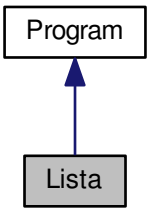
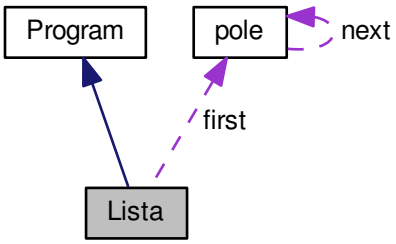


Diagram współpracy dla Lista:



Metody publiczne

- Lista ()

  Konstruktor bezparametryczny.
- void push (int x)

  Metoda push.
- void pop ()

  Procedura pop.
- int size ()

  Metoda size.
- bool wykonaj_program (char ∗nazwa_pliku, int ilosc_danych)

  Metoda wykonaj_program.
- void wyczysc_dane (int ile)

  Metoda wyczysc_dane.

Atrybuty prywatne

- pole * first

Dodatkowe Dziedziczone Składowe

4.3.1 Opis szczegółowy

Definicja w linii 25 pliku lista.hh.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 Lista::Lista ( ) [inline]

Ustawia poczatek listy na NULL

Definicja w linii 33 pliku lista.hh.

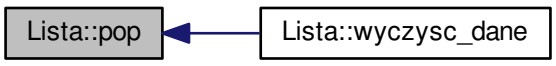
4.3.3 Dokumentacja funkcji składowych

4.3.3.1 void Lista::pop ( )

Usuwa ostatni element listy.

Definicja w linii 22 pliku lista.cpp.

Oto graf wywoływań tej funkcji:



4.3.3.2 void Lista::push (int *x*)

Dodaje podana wartosc na koniec listy.

Parametry

in	*x*	Wartosc, ktora chcemy dodac na koniec listy.

Definicja w linii 7 pliku lista.cpp.

Oto graf wywoływań tej funkcji:

4.3.3.3 int Lista::size ( )

Daje informacje o rozmiarze listy (liczbie jej elementow).

Zwraca

Rozmiar listy (liczba jej elementow)

Definicja w linii 44 pliku lista.cpp.

4.3.3.4 void Lista::wyczysc_dane (int *ile*) [virtual]

Usuwa zadana ilosc elementow listy za pomoca metody pop

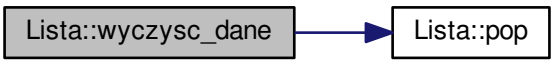
Parametry

in	*ile*	Liczba elementow, ktore chcemy usunac.

Implementuje Program.

Definicja w linii 74 pliku lista.cpp.

Oto graf wywołań dla tej funkcji:



4.3.3.5 bool Lista::wykonaj_program (char * *nazwa_pliku,* int *ilosc_danych*) [virtual]

Wykonuje zadany program - dodanie zadanej ilosci danych z pliku do listy za pomoca metody push()

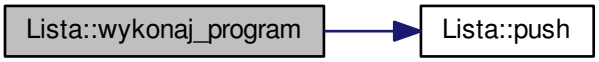
Zwracane wartości

TRUE	Poprawnie wykonano program
FALSE	Nie wczytano danych

Implementuje Program.

Definicja w linii 58 pliku lista.cpp.

Oto graf wywołań dla tej funkcji:



4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 pole* Lista::first [private]

Definicja w linii 26 pliku lista.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- lista.hh
- lista.cpp

4.4 Dokumentacja klasy Lista_tab

`#include <lista_tab.hh>`

Diagram dziedziczenia dla Lista_tab

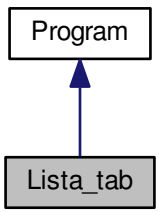
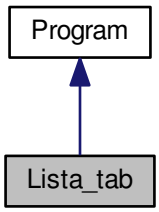


Diagram współpracy dla Lista_tab:



Metody publiczne

- Lista_tab ()

  Konstruktor bezparametryczny.
- ~Lista_tab ()

  Destruktor.
- void push (int x)

  Metoda push.
- void pop ()

  Procedura pop.
- int size ()

  Metoda size.
- bool wykonaj_program (char *nazwa_pliku, int ilosc_danych)

  Metoda wykonaj_program.
- void wyczysc_dane (int ile)

  Metoda wyczysc_dane.

Atrybuty prywatne

- int rozmiar

  Aktualny rozmiar tablicy.
- int iterator

  Iterator, zawierajacy informacje o liczbie elementow w tablicy.
- int * tab

  Wskaznik na dynamicznie alokowana tablice z danymi.

Dodatkowe Dziedziczone Składowe

4.4.1 Opis szczegółowy

Definicja w linii 12 pliku lista_tab.hh.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 Lista_tab::Lista_tab ( ) [inline]

Ustawia poczatek listy na NULL

Definicja w linii 31 pliku lista_tab.hh.

4.4.2.2 Lista_tab::~Lista_tab ( ) [inline]

Usuwa dynamicznie utworzona tablice danych oraz przypisuje wskaznikowi wartosc NULL.

Definicja w linii 43 pliku lista_tab.hh.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 void Lista_tab::pop ( )

Usuwa ostatni element listy.

Definicja w linii 54 pliku lista_tab.cpp.

4.4.3.2 void Lista_tab::push (int *x*)

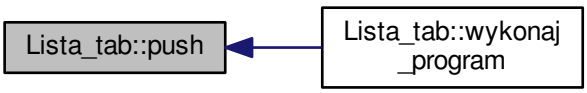
Dodaje podana wartosc na koniec listy.

Parametry

in	*x*	Wartosc, ktora chcemy dodac na koniec listy.

Definicja w linii 8 pliku lista_tab.cpp.

Oto graf wywoływań tej funkcji:



4.4.3.3 int Lista_tab::size ( )

Daje informacje o rozmiarze listy (liczbie jej elementow).

Zwraca

Rozmiar listy (liczba jej elementow)

Definicja w linii 71 pliku lista_tab.cpp.

4.4.3.4 void Lista_tab::wyczysc_dane (int *ile*) [virtual]

Usuwa zadana ilosc elementow listy za pomoca metody pop

Parametry

in	*ile*	Liczba elementow, ktore chcemy usunac.

Implementuje Program.

Definicja w linii 91 pliku lista_tab.cpp.

4.4.3.5 bool Lista_tab::wykonaj_program (char * *nazwa_pliku,* int *ilosc_danych*) [virtual]

Wykonuje zadany program - dodanie zadanej ilosci danych z pliku do listy za pomoca metody push()

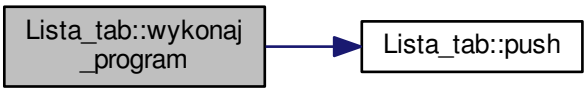
Zwracane wartości

TRUE	Poprawnie wykonano program
FALSE	Nie wczytano danych

Implementuje Program.

Definicja w linii 75 pliku lista_tab.cpp.

Oto graf wywołań dla tej funkcji:



4.4.4 Dokumentacja atrybutów składowych

4.4.4.1 int Lista_tab::iterator [private]

Definicja w linii 20 pliku lista_tab.hh.

4.4.4.2 int Lista_tab::rozmiar [private]

Definicja w linii 16 pliku lista_tab.hh.

4.4.4.3 int* Lista_tab::tab [private]

Definicja w linii 24 pliku lista_tab.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

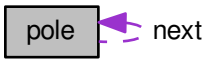
- lista_tab.hh
- lista_tab.cpp

4.5 Dokumentacja struktury pole

Struktura pole.

`#include <lista.hh>`

Diagram współpracy dla pole:



Metody publiczne

- pole ()

Atrybuty publiczne

- int wartosc
- pole ∗ next

4.5.1 Opis szczegółowy

Jest to struktura skladowa klasy Lista, zawierajaca przechowywana wartosc oraz wskaznik na zmienna typu pole.

Definicja w linii 19 pliku lista.hh.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 pole::pole ( ) [inline]

Definicja w linii 22 pliku lista.hh.

4.5.3 Dokumentacja atrybutów składowych

4.5.3.1 pole∗ pole::next

Definicja w linii 21 pliku lista.hh.

4.5.3.2 int pole::wartosc

Definicja w linii 20 pliku lista.hh.

Dokumentacja dla tej struktury została wygenerowana z pliku:

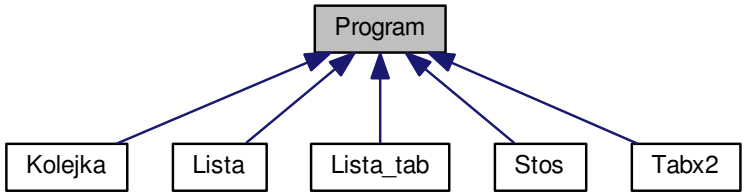
- lista.hh

4.6 Dokumentacja klasy Program

Modeluje klase Program.

`#include <program.hh>`

Diagram dziedziczenia dla Program



Metody publiczne

- int getRozmiar_tab ()
  Akcesor getRozmiar_tab.
- Program ()
  Konstruktor bezparametryczny.
- ~Program ()
  Destruktor.
- bool wczytaj_dane (char *nazwa_pliku)
  Metoda wczytaj_dane.
- bool wczytaj_dane (char *nazwa_pliku, int ile_danych)
  Metoda wczytaj_dane.
- bool zapisz_dane (char *nazwa_pliku)
- void wyswietl_dane ()
  Procedura wyswietl_dane.
- virtual bool wykonaj_program ()
  Wirtualna metoda wykonaj_program.
- virtual bool wykonaj_program (char *nazwa_pliku, int ilosc_danych)=0
- virtual void wyczysc_dane (int ile)=0

Atrybuty chronione

- int rozmiar_tab
  Zmiena rozmiar_tab.
- int * tab
  Zmienna tablica.
- ifstream plik_we
  Zmienna plik_we.
- ofstream plik_wy
  Zmienna plik_wy.

4.6.1 Opis szczegółowy

Klasa Program zawiera zmienne oraz metody wspolne dla wszystkich programow. Sa one zwiazane z przechowywaniem i obsluga danych.

Definicja w linii 22 pliku program.hh.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 Program::Program ( ) [inline]

Przypisuje domyslna wartosc 0 dla rozmiaru tablicy danych oraz NULL dla wskaznika.

Definicja w linii 71 pliku program.hh.

4.6.2.2 Program::~Program ( ) [inline]

Usuwa dynamicznie utworzona tablice danych oraz przypisuje wskaznikowi wartosc NULL.

Definicja w linii 79 pliku program.hh.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 int Program::getRozmiar_tab ( ) [inline]

Metoda dajaca mozliwosc odczytu rozmiaru tablicy.

Definicja w linii 63 pliku program.hh.

4.6.3.2 bool Program::wczytaj_dane (char * *nazwa_pliku*)

Wczytuje dane z pliku. W pierwszej linii pliku musi znajdowac sie informacja o ilosci wczytywanych danych, dane w kolejnych liniach: ilosc_danych dana1 dana2 ...

Parametry

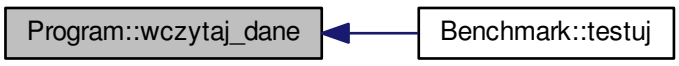
in	*nazwa_pliku*	Wskaznik do nazwy pliku do wczytania.

Zwracane wartości

TRUE	Poprawnie wczytano plik.
FALSE	Blad podczas wczytywania pliku.

Definicja w linii 8 pliku program.cpp.

Oto graf wywoływań tej funkcji:



4.6.3.3 bool Program::wczytaj_dane (char * *nazwa_pliku,* int *ile_danych*)

Wczytuje okreslona liczbe danych z pliku. W pierwszej linii pliku musi znajdowac sie informacja o ilosci wczytywanych danych, dane w kolejnych liniach: ilosc_danych dana1 dana2 ...

Parametry

in	*nazwa_pliku*	Wskaznik do nazwy pliku do wczytania.
in	*ile_danych*	Ilosc danych, jakie chcemy wczytac.

Zwracane wartości

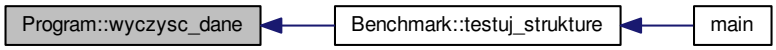
TRUE	Poprawnie wczytano plik.
FALSE	Blad podczas wczytywania pliku.

Definicja w linii 26 pliku program.cpp.

4.6.3.4 virtual void Program::wyczysc_dane (int *ile*) [pure virtual]

Implementowany w Lista_tab, Lista, Kolejka i Stos.

Oto graf wywoływań tej funkcji:



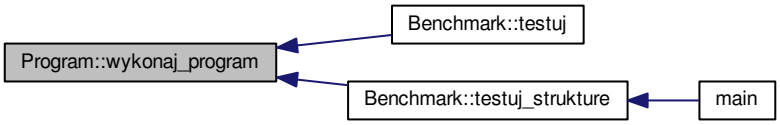
4.6.3.5 bool Program::wykonaj_program ( ) [virtual]

Wykonuje program na zadanej liczbie danych.

Reimplementowana w Tabx2.

Definicja w linii 67 pliku program.cpp.

Oto graf wywoływań tej funkcji:



4.6.3.6 virtual bool Program::wykonaj_program (char * *nazwa_pliku,* int *ilosc_danych*) [pure virtual]

Implementowany w Lista_tab, Lista, Kolejka i Stos.

4.6.3.7 void Program::wyswietl_dane ( )

Wypisuje wczytane dane jedna pod druga na standardowy strumien wyjscia.

Definicja w linii 62 pliku program.cpp.

4.6.3.8 bool Program::zapisz_dane (char * *nazwa_pliku*)

Metoda zapisz_dane

Zapisuje przetworzone dane do pliku. W pierwszej linijce zamieszcza informacje o ilosci danych, w kolejnych liniach pojedyncze dane: ilosc_danych dana1 dana2 ...

Parametry

in	*nazwa_pliku*	Wskaznik do nazwy pliku do zapisu.

Zwracane wartości

TRUE	Poprawnie zapisano plik.
FALSE	Blad podczas zapisu pliku.

Definicja w linii 47 pliku program.cpp.

4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 ifstream Program::plik_we [protected]

Zmienna przechowujaca strumien wejsciowy do otwartego pliku z wczytywanymi danymi.

Definicja w linii 47 pliku program.hh.

4.6.4.2 ofstream Program::plik_wy [protected]

Zmienna przechowujaca strumien wyjsciowy do tworzonego pliku z danymi po przetworzeniu.

Definicja w linii 55 pliku program.hh.

4.6.4.3 int Program::rozmiar_tab [protected]

Zmienna przechowujaca informacje o ilosci wczytanych danych, ktora rowna jest dlugosci utworzonej tablicy dynamicznej (wskazywanej wskaznikiem tab).

Definicja w linii 31 pliku program.hh.

4.6.4.4 int* Program::tab [protected]

Zamienna wskaznikowa wskazujaca na dynamicznie tworzona tablice z danymi.

Definicja w linii 39 pliku program.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- program.hh
- program.cpp

4.7 Dokumentacja klasy Stos

```
#include <stos.hh>
```

Diagram dziedziczenia dla Stos

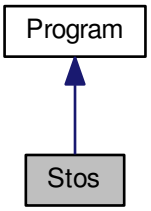
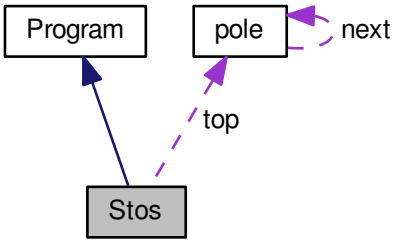


Diagram współpracy dla Stos:



Metody publiczne

- Stos ()

  Konstruktor bezparametryczny.
- void push (int x)

  Metoda push.
- void pop ()

  Procedura pop.
- int size ()

  Metoda size.
- bool wykonaj_program (char ∗nazwa_pliku, int ilosc_danych)

  Metoda wykonaj_program.
- void wyczysc_dane (int ile)

  Metoda wyczysc_dane.

Atrybuty prywatne

- pole ∗ top

Dodatkowe Dziedziczone Składowe

4.7.1 Opis szczegółowy

Definicja w linii 13 pliku stos.hh.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 Stos::Stos ( ) [inline]

Ustawia wierzcholek stosu na NULL

Definicja w linii 21 pliku stos.hh.

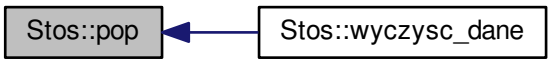
4.7.3 Dokumentacja funkcji składowych

4.7.3.1 void Stos::pop ( )

Usuwa element znajdujacy sie na szczycie stosu (ostatnio dodany).

Definicja w linii 14 pliku stos.cpp.

Oto graf wywoływań tej funkcji:



4.7.3.2 void Stos::push (int *x*)

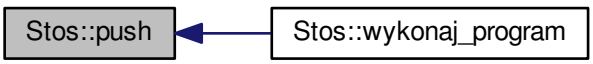
Dodaje podana wartosc na wierzcholek stosu.

Parametry

in	*x*	Wartosc, ktora chcemy dodac na wierzcholek stosu.

Definicja w linii 7 pliku stos.cpp.

Oto graf wywoływań tej funkcji:



4.7.3.3 int Stos::size ( )

Daje informacje o rozmiarze stosu (liczbie jego elementow).

Zwraca

Rozmiar stosu (liczba jego elementow)

Definicja w linii 24 pliku stos.cpp.

4.7.3.4 void Stos::wyczysc_dane (int *ile*) [virtual]

Usuwa zadana ilosc elementow stosu za pomoca metody pop

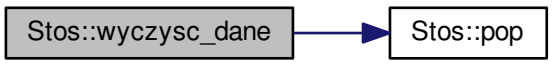
Parametry

in	*ile*	Liczba elementow, ktore chcemy usunac.

Implementuje Program.

Definicja w linii 55 pliku stos.cpp.

Oto graf wywołań dla tej funkcji:



4.7.3.5 bool Stos::wykonaj_program (char * *nazwa_pliku,* int *ilosc_danych*) [virtual]

Wykonuje zadany program - dodanie zadanej ilosci danych z pliku do stosu za pomoca metody push()

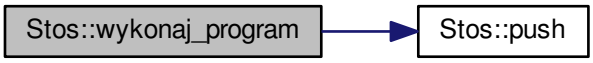
Zwracane wartości

TRUE	Poprawnie wykonano program
FALSE	Nie wczytano danych

Implementuje Program.

Definicja w linii 39 pliku stos.cpp.

Oto graf wywołań dla tej funkcji:



4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 pole* Stos::top [private]

Definicja w linii 14 pliku stos.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- stos.hh
- stos.cpp

4.8 Dokumentacja klasy Tabx2

`#include <tabx2.hh>`

Diagram dziedziczenia dla Tabx2

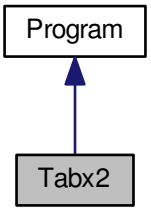
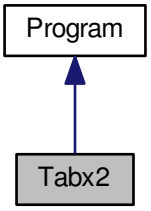


Diagram współpracy dla Tabx2:



Metody publiczne

- virtual bool wykonaj_program ()

  Metoda wirtualna wykonaj_program.

Dodatkowe Dziedziczone Składowe

4.8.1 Opis szczegółowy

Definicja w linii 18 pliku tabx2.hh.

4.8.2 Dokumentacja funkcji składowych

4.8.2.1 bool Tabx2::wykonaj_program ( ) [virtual]

Dokonuje przemnozenia przez 2 wszystkich danych znajdujacych sie w tablicy wskazywanej przez tab.

Zwracane wartości

TRUE	Poprawnie dokonano mnozenia wszystkich liczb
FALSE	Rozmiar tablicy danych wynosi 0

Reimplementowana z Program.

Definicja w linii 7 pliku tabx2.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- tabx2.hh
- tabx2.cpp

Rozdział 5

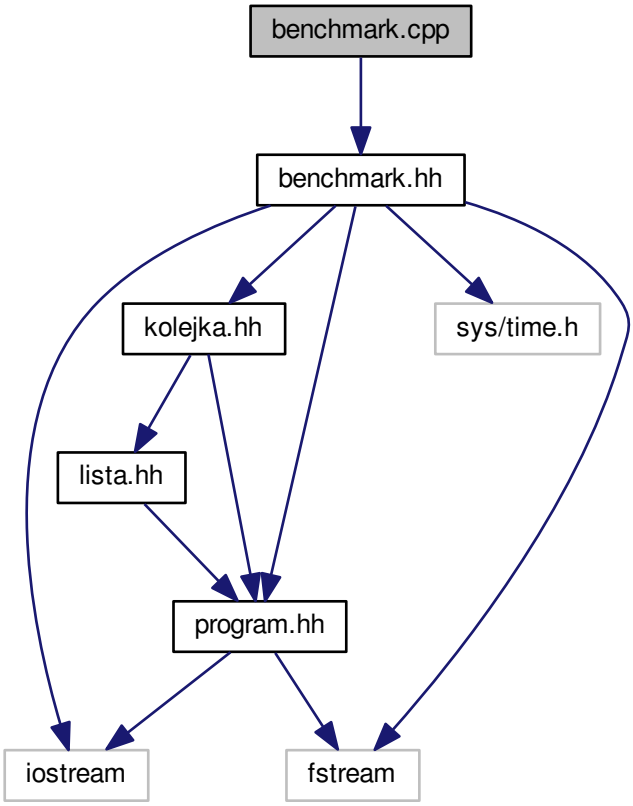
Dokumentacja plików

5.1 Dokumentacja pliku benchmark.cpp

Plik zawiera metody klasy Benchmark.

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

Wykres zależności załączania dla benchmark.cpp:



5.2 benchmark.cpp

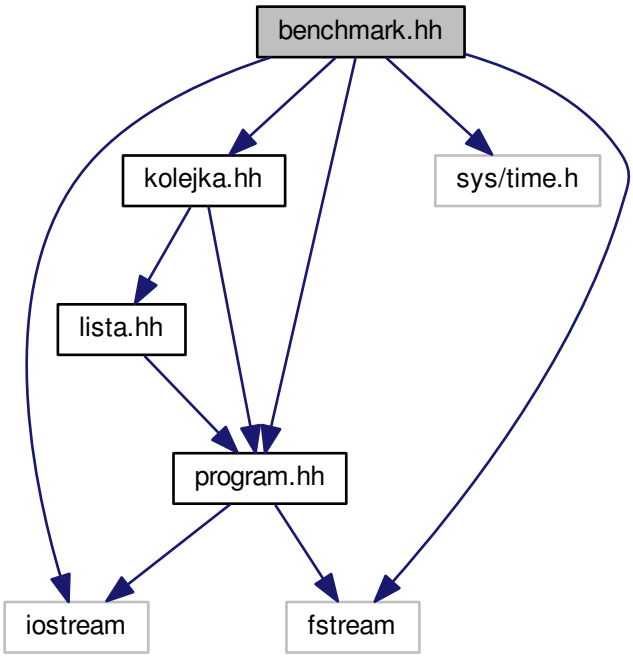
```
00001 #include "benchmark.hh"
00002
00007 void Benchmark::rozpocznij_pomiar(){
00008   gettimeofday(&t1, NULL);
00009 }
00010
00011 void Benchmark::zakoncz_pomiar(){
00012   gettimeofday(&t2, NULL);
00013   czas_pomiaru = (t2.tv_sec - t1.tv_sec) * 1000.0;      // sec to ms
00014   czas_pomiaru += (t2.tv_usec - t1.tv_usec) / 1000.0;   // us to ms
00015 }
00016
00017 double Benchmark::testuj(Program &program, char* dane, int ilosc_danych, int
      ilosc_testow){
00018   double suma=0;
00019   double srednia=0;
00020   ofstream wyniki;
00021   wyniki.open("wyniki.csv",ios::app);
00022
00023   if(program.wczytaj_dane(dane,ilosc_danych)==false){
00024       cerr<<"Niewystarczajaca ilosc danych!"<<endl;
00025       return 0;
00026     }
00027   //char* dane_wy = (char*)"dane_wy.dat"; //do zapisu do pliku
00028   rozpocznij_pomiar();
00029   program.wykonaj_program();
00030   //program.zapisz_dane(dane_wy);//zapisywanie wynikow do pliku
00031   zakoncz_pomiar();
00032   suma+=czas_pomiaru;
00033   for(int i=1;i<ilosc_testow;i++){
00034     //program.wczytaj_dane(dane,ilosc_danych); //zawsze dane od poczatku
00035     rozpocznij_pomiar();
00036     program.wykonaj_program();
00037     zakoncz_pomiar();
00038     suma+=czas_pomiaru;
00039   }
00040   srednia=suma/(ilosc_testow);
00041   wyniki<<endl<<ilosc_danych<<","<<srednia;
00042   wyniki.close();
00043   return srednia;
00044 }
00045
00046 double Benchmark::testuj_strukture(Program &program,char* dane, int
      ilosc_danych, int ilosc_testow){
00047   double suma=0;
00048   double srednia=0;
00049   ofstream wyniki;
00050
00051   rozpocznij_pomiar();
00052   program.wykonaj_program(dane,ilosc_danych);
00053   zakoncz_pomiar();
00054   program.wyczysc_dane(ilosc_danych);
00055   suma+=czas_pomiaru;
00056   for(int i=1;i<ilosc_testow;i++){
00057     rozpocznij_pomiar();
00058     program.wykonaj_program(dane,ilosc_danych);
00059     zakoncz_pomiar();
00060     program.wyczysc_dane(ilosc_danych);
00061     suma+=czas_pomiaru;
00062   }
00063   srednia=suma/(ilosc_testow);
00064
00065   wyniki.open("wyniki.csv",ios::app);
00066   wyniki<<endl<<ilosc_danych<<","<<srednia;
00067   wyniki.close();
00068   return srednia;
00069 }
```

5.3 Dokumentacja pliku benchmark.hh

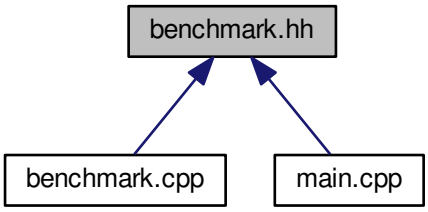
Definicja klasy Benchmark.

```
#include <iostream>
#include "program.hh"
#include <sys/time.h>
#include <fstream>
#include "kolejka.hh"
```

Wykres zależności załączania dla benchmark.hh:



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



Komponenty

- class Benchmark

  Klasa Benchmark.

5.4 benchmark.hh

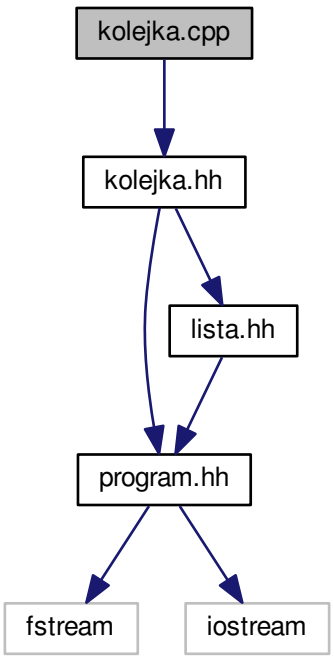
```
00001 //benchmark.hh
```

```
00002
00003 #ifndef BENCHMARK_HH
00004 #define BENCHMARK_HH
00005
00006 #include <iostream>
00007 #include "program.hh"
00008 #include <sys/time.h>
00009 #include <fstream>
00010 #include "kolejka.hh"
00016 using namespace std;
00017
00023 class Benchmark{
00024 private:
00030   timeval t1, t2;
00031
00037   double czas_pomiaru;
00038
00039 public:
00045   void rozpocznij_pomiar();
00046
00052   void zakoncz_pomiar();
00053
00066   double testuj(Program &program,char* dane, int ilosc_danych, int ilosc_testow);
00067
00081   double testuj_strukture(Program &program,char* dane, int ilosc_danych, int ilosc_testow);
00082 };
00083
00084 #endif
```

5.5 Dokumentacja pliku kolejka.cpp

Zawiera definicje metod klasy Kolejka.

`#include "kolejka.hh"`
Wykres zależności załączania dla kolejka.cpp:



5.6 kolejka.cpp

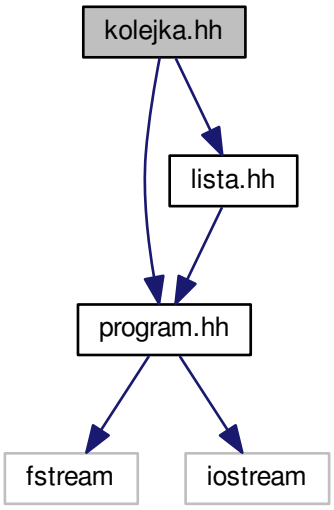
```
00001 //kolejka.cpp
00002 #include "kolejka.hh"
00007 void Kolejka::push(int x){
00008   pole *nowe = new pole;
00009   nowe->wartosc = x;
00010   if(head == NULL){
00011     head = nowe;
00012     tail = nowe;
00013   }
00014   else{
00015     tail->next = nowe;
00016     tail = nowe;
00017   }
00018 }
00019
00020 void Kolejka::pop(){
00021   if(head == NULL)
00022     cerr<<"Kolejka jest pusta!"<<endl;
00023   else{
00024     pole *wsk = head;
00025     head = head->next;
00026     if(head == NULL) tail = NULL;
00027     delete wsk;
00028   }
00029 }
00030
00031 int Kolejka::size(){
00032   if(head == NULL)
00033     return 0;
00034   else{
00035     pole *wsk = head;
00036     int i=1;
00037     while(wsk->next){
00038       wsk=wsk->next;
00039       i++;
00040     }
00041     return i;
00042   }
00043 }
00044
00045 bool Kolejka::wykonaj_program(char* nazwa_pliku, int ilosc_danych){
00046   int tmp;
00047   plik_we.open(nazwa_pliku);
00048   if(plik_we.good()==false){
00049     cerr<<"Blad odczytu pliku!"<<endl;
00050     return false;
00051   }
00052   plik_we >> rozmiar_tab;
00053   for(int i=0;i<ilosc_danych;i++){
00054     plik_we >> tmp;
00055     push(tmp);
00056   }
00057   plik_we.close();
00058   return true;
00059 }
00060
00061 void Kolejka::wyczysc_dane(int ile){
00062   for(int i=0;i<ile;i++)
00063     pop();
00064 }
```

5.7 Dokumentacja pliku kolejka.hh

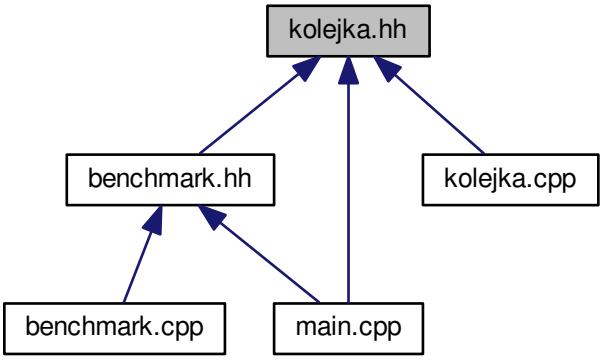
Definicja klasy Kolejka.

```
#include "program.hh"
#include "lista.hh"
```

Wykres zależności załączania dla kolejka.hh:



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



Komponenty

- class Kolejka

5.8 kolejka.hh

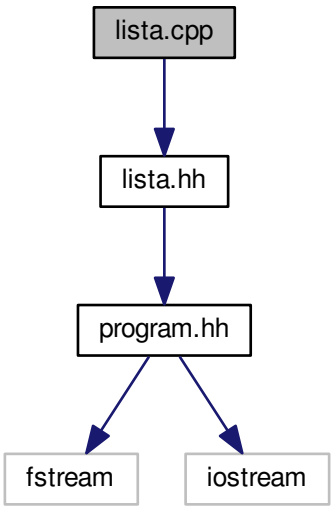
```
00001 #ifndef KOLEJKA_HH
00002 #define KOLEJKA_HH
```

```
00003
00004 #include "program.hh"
00005 #include "lista.hh"
00011 class Kolejka: public Program{
00012   pole *head;
00013   pole *tail;
00014
00015 public:
00021   Kolejka(){
00022     head = NULL;
00023     tail = NULL;
00024   }
00032   void push(int x);
00038   void pop();
00046   int size();
00047
00057   bool wykonaj_program(char* nazwa_pliku,int ilosc_danych);
00058
00066   void wyczysc_dane(int ile);
00067 };
00068
00069 #endif
```

5.9 Dokumentacja pliku lista.cpp

Zawiera definicje metod klasy Lista.

`#include "lista.hh"`
Wykres zależności załączania dla lista.cpp:



5.10 lista.cpp

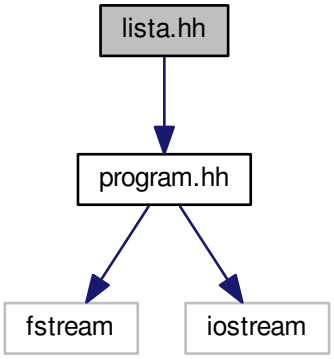
```
00001 //lista.cpp
00002 #include "lista.hh"
00003
00007 void Lista::push(int x){
00008   pole *nowe = new pole;
00009   nowe->wartosc = x;
00010   if(first == NULL){
00011     first = nowe;
00012   }
```

```
00013   else{
00014     pole *wsk = first;
00015     while(wsk->next)
00016       wsk=wsk->next;
00017     wsk->next = nowe;
00018     nowe->next = NULL;
00019   }
00020 }
00021
00022 void Lista::pop(){
00023   if(first == NULL){
00024     cerr<<"Lista jest pusta!"<<endl;
00025   }
00026   else{
00027     pole *wsk = first;
00028     pole *prev = NULL;
00029     while(wsk->next){
00030       prev=wsk;
00031       wsk=wsk->next;
00032     }
00033     if(prev==NULL){
00034       delete wsk;
00035       wsk = NULL;
00036     }
00037     else{
00038       prev->next = NULL;
00039       delete wsk;
00040     }
00041   }
00042 }
00043
00044 int Lista::size(){
00045   if(first == NULL)
00046     return 0;
00047   else{
00048     pole *wsk = first;
00049     int i=1;
00050     while(wsk->next){
00051       wsk=wsk->next;
00052       i++;
00053     }
00054     return i;
00055   }
00056 }
00057
00058 bool Lista::wykonaj_program(char* nazwa_pliku, int ilosc_danych){
00059   int tmp;
00060   plik_we.open(nazwa_pliku);
00061   if(plik_we.good()==false){
00062     cerr<<"Blad odczytu pliku!"<<endl;
00063     return false;
00064   }
00065   plik_we >> rozmiar_tab;
00066   for(int i=0;i<ilosc_danych;i++){
00067     plik_we >> tmp;
00068     push(tmp);
00069   }
00070   plik_we.close();
00071   return true;
00072 }
00073
00074 void Lista::wyczysc_dane(int ile){
00075   for(int i=0;i<ile;i++)
00076     pop();
00077 }
```

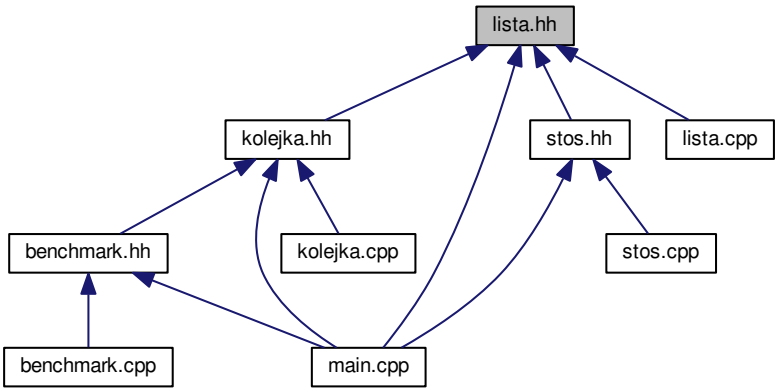
5.11 Dokumentacja pliku lista.hh

Definicja klasy Lista.

`#include "program.hh"`
Wykres zależności załączania dla lista.hh:



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



Komponenty

- struct pole

  Struktura pole.
- class Lista

5.12 lista.hh

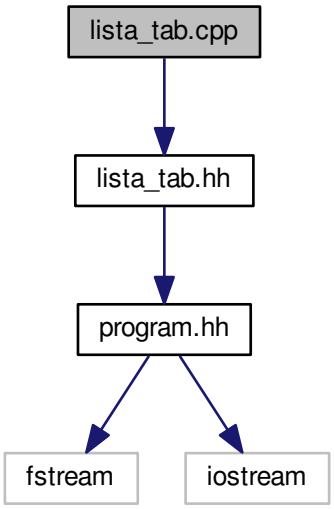
```
00001 //lista.hh
00002 #ifndef LISTA_HH
00003 #define LISTA_HH
00004
00005 #include "program.hh"
```

```
00006
00019 struct pole{
00020   int wartosc;
00021   pole *next;
00022   pole(){wartosc=0; next=NULL;}
00023 };
00024
00025 class Lista: public Program{
00026   pole *first;
00027 public:
00033  Lista(){
00034     first = NULL;
00035   }
00043   void push(int x);
00049   void pop();
00057   int size();
00058
00068   bool wykonaj_program(char* nazwa_pliku,int ilosc_danych);
00069
00077   void wyczysc_dane(int ile);
00078 };
00079
00080 #endif
```

5.13 Dokumentacja pliku lista_tab.cpp

Zawiera definicje metod klasy Lista.

`#include "lista_tab.hh"`
Wykres zależności załączania dla lista_tab.cpp:



5.14 lista_tab.cpp

```
00001 //lista_tab.cpp
00002 #include "lista_tab.hh"
00003
00007 // ZWIEKSZA SIE O 1
00008 void Lista_tab::push(int x){
00009   if(rozmiar==0){
00010     tab = new int [1];
00011     rozmiar=1;
```

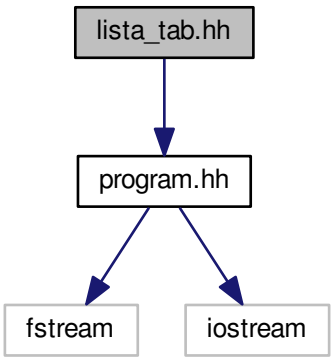
```
00012     iterator=0;
00013     tab[iterator]=x;
00014   }
00015   else{
00016     if(iterator<rozmiar-1){
00017       tab[++iterator]=x;
00018     }
00019     else if(iterator>=rozmiar-1){
00020       int *tmp = new int[rozmiar+1];
00021       for(int i=0;i<=iterator;i++)
00022     tmp[i] = tab[i];
00023       delete [] tab;
00024       tab = tmp;
00025       tab[++iterator]=x;
00026       rozmiar++;
00027     }
00028   }
00029 }/*
00030    void Lista_tab::push(int x){
00031    if(rozmiar==0){
00032    tab = new int [1];
00033     rozmiar=1;
00034     iterator=0;
00035     tab[iterator]=x;
00036   }
00037   else{
00038     if(iterator<rozmiar-1){
00039       tab[++iterator]=x;
00040     }
00041     else if(iterator>=rozmiar-1){
00042       int *tmp = new int[2*rozmiar];
00043       for(int i=0;i<=iterator;i++)
00044     tmp[i] = tab[i];
00045       delete [] tab;
00046       tab = tmp;
00047       tab[++iterator]=x;
00048       rozmiar*=2;
00049     }
00050   }
00051   }*/
00052
00053
00054 void Lista_tab::pop(){
00055   if(rozmiar == 0){
00056     cerr<<"Lista jest pusta!"<<endl;
00057   }
00058   iterator--;
00059   if(iterator<0.25*(rozmiar-1)){
00060     int *tmp = new int[iterator+1];
00061     for(int i=0;i<=iterator;i++){
00062       tmp[i]=tab[i];
00063     }
00064     delete [] tab;
00065     tab = tmp;
00066     rozmiar = iterator+1;
00067   }
00068 }
00069
00070
00071 int Lista_tab::size(){
00072   return rozmiar;
00073 }
00074
00075 bool Lista_tab::wykonaj_program(char* nazwa_pliku, int ilosc_danych){
00076   int tmp;
00077   plik_we.open(nazwa_pliku);
00078   if(plik_we.good()==false){
00079     cerr<<"Blad odczytu pliku!"<<endl;
00080     return false;
00081   }
00082   plik_we >> rozmiar_tab;  //pozbyc sie pierwszej liczby z pliku z danymi
00083   for(int i=0;i<ilosc_danych;i++){
00084     plik_we >> tmp;
00085     push(tmp);
00086   }
00087   plik_we.close();
00088   return true;
00089 }
00090
00091 void Lista_tab::wyczysc_dane(int ile){
00092   delete [] tab;
00093   rozmiar = 0;
00094   iterator = 0;
00095   tab = NULL;
00096 }
00097
```

5.15 Dokumentacja pliku lista_tab.hh

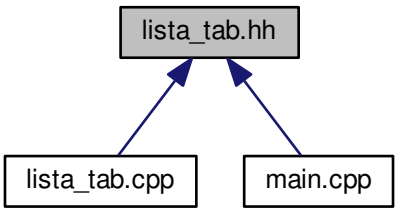
Definicja klasy Lista_tab.

```
#include "program.hh"
```

Wykres zależności załączania dla lista_tab.hh:



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



Komponenty

- class Lista_tab

Definicje

- #define LISTA__TAB_HH

5.15.1 Dokumentacja definicji

5.15.1.1 #define LISTA__TAB_HH

Definicja w linii 3 pliku lista_tab.hh.

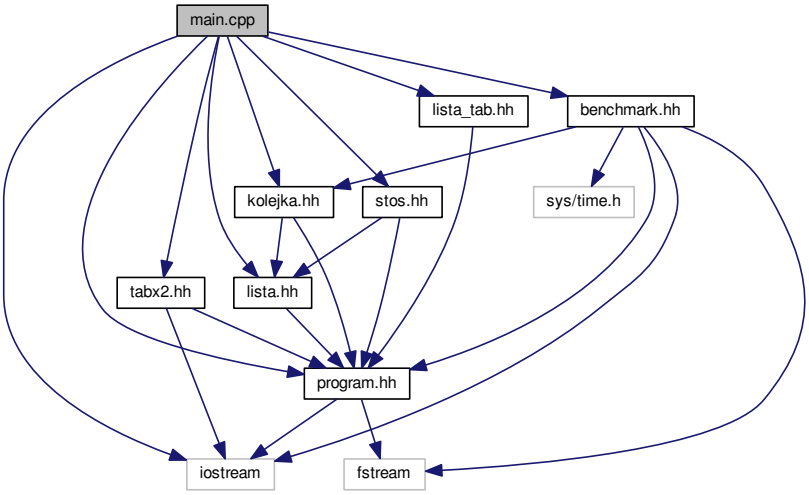
5.16 lista_tab.hh

```
00001 //lista_tab.hh
00002 #ifndef LISTA_TAB_HH
00003 #define LISTA__TAB_HH
00004
00005 #include "program.hh"
00006
00012 class Lista_tab: public Program{
00016   int rozmiar;
00020   int iterator;
00024   int *tab;
00025 public:
00031  Lista_tab(){
00032     tab = NULL;
00033     rozmiar = 0;
00034     iterator = 0;
00035   }
00036
00043   ~Lista_tab(){delete[] tab; tab=NULL; rozmiar=0;
      iterator=0;}
00044
00052   void push(int x);
00058   void pop();
00066   int size();
00067
00077   bool wykonaj_program(char* nazwa_pliku,int ilosc_danych);
00078
00086   void wyczysc_dane(int ile);
00087 };
00088
00089 #endif
```

5.17 Dokumentacja pliku main.cpp

```
#include <iostream>
#include "program.hh"
#include "tabx2.hh"
#include "benchmark.hh"
#include "stos.hh"
#include "kolejka.hh"
#include "lista.hh"
#include "lista_tab.hh"
```

Wykres zależności załączania dla main.cpp:



Funkcje

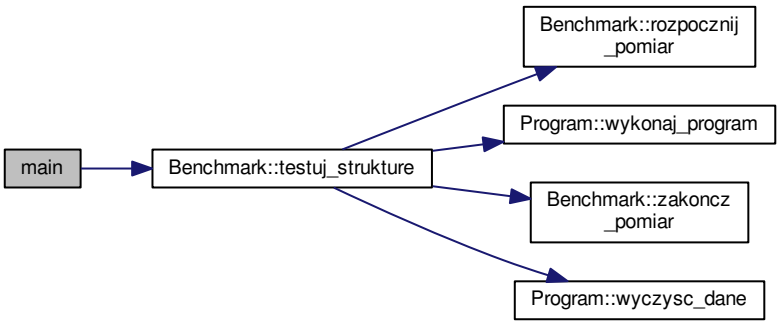
- int main ()

5.17.1 Dokumentacja funkcji

5.17.1.1 int main ( )

Definicja w linii 13 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



5.18 main.cpp

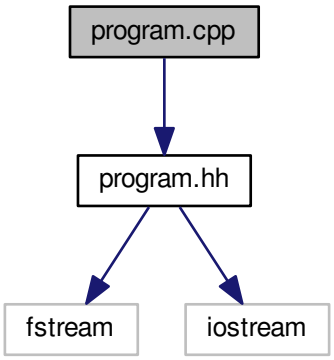
```
00001 //main.cpp
00002 #include <iostream>
```

```
00003 #include "program.hh"
00004 #include "tabx2.hh"
00005 #include "benchmark.hh"
00006 #include "stos.hh"
00007 #include "kolejka.hh"
00008 #include "lista.hh"
00009 #include "lista_tab.hh"
00010
00011 using namespace std;
00012
00013 int main(){
00014
00015   Lista_tab a;
00016   Benchmark b;
00017   char* dane = (char*)"dane.dat";
00018   int ilosc_testow = 10;
00019
00020   for(int ilosc_danych=1; ilosc_danych<=100000000;ilosc_danych*=2){
00021     cout << b.testuj_strukture(a,dane,ilosc_danych,ilosc_testow) << endl;
00022   }
00023
00024   return 0;
00025 }
```

5.19 Dokumentacja pliku program.cpp

Plik zawiera metody klasy Program.

`#include "program.hh"`
Wykres zależności załączania dla program.cpp:



5.20 program.cpp

```
00001 //program.cpp
00002 #include "program.hh"
00003
00004 using namespace std;
00008 bool Program::wczytaj_dane(char* nazwa_pliku){
00009   if(plik_we.good()==true)
00010     plik_we.close();
00011   plik_we.open(nazwa_pliku);
00012   if(plik_we.good()==false){
00013     cerr<<"Blad odczytu pliku!";
00014     return false;
00015   }
00016   plik_we >> rozmiar_tab;
00017   tab=new int [rozmiar_tab];
00018   int i=0;
```

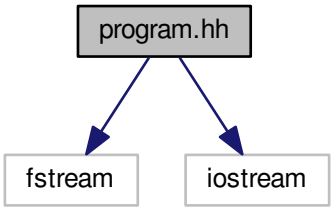
```
00019   while(plik_we >> tab[i]){
00020     i++;
00021   }
00022   plik_we.close();
00023   return true;
00024 }
00025
00026 bool Program::wczytaj_dane(char* nazwa_pliku, int ile_danych){
00027   if(plik_we.good()==true)
00028     plik_we.close();
00029   plik_we.open(nazwa_pliku);
00030   if(plik_we.good()==false){
00031     cerr<<"Blad odczytu pliku!"<<endl;
00032     return false;
00033   }
00034   plik_we >> rozmiar_tab;
00035   if(ile_danych>rozmiar_tab){
00036     plik_we.close();
00037     return false;
00038   }
00039   rozmiar_tab=ile_danych;
00040   tab=new int [ile_danych];
00041   for(int i=0;i<ile_danych;i++)
00042     plik_we>>tab[i];
00043   plik_we.close();
00044   return true;
00045 }
00046
00047 bool Program::zapisz_dane(char* nazwa_pliku){
00048   if(plik_wy.good()==true)
00049     plik_wy.close();
00050   plik_wy.open(nazwa_pliku);
00051   if(plik_wy.good()==false){
00052     cerr<<"Blad odczytu pliku!";
00053     return false;
00054   }
00055   plik_wy << rozmiar_tab << endl;
00056   for(int i=0;i<rozmiar_tab;i++)
00057     plik_wy << tab[i] << endl;
00058   plik_wy.close();
00059   return true;
00060 }
00061
00062 void Program::wyswietl_dane(){
00063   for(int i=0;i<rozmiar_tab;i++)
00064     cout<<tab[i]<<endl;
00065 }
00066
00067 bool Program::wykonaj_program(){
00068   cerr<<"Nie wybrano programu do wykonania!"<<endl;
00069   return false;
00070 }
```

5.21 Dokumentacja pliku program.hh

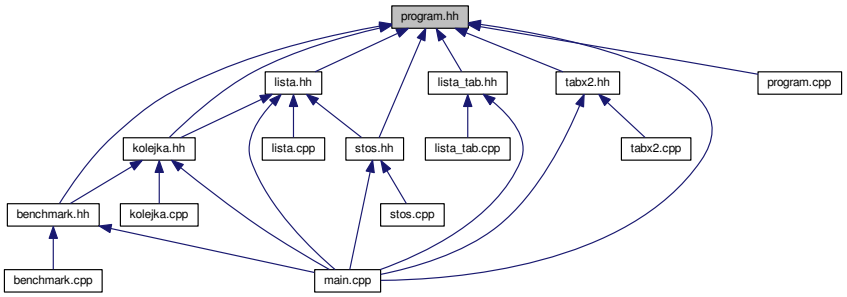
Definicja klasy Program.

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

Wykres zależności załączania dla program.hh:



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



Komponenty

- class Program

  Modeluje klase Program.

5.22 program.hh

```
00001 //program.hh
00002
00003 #ifndef PROGRAM_HH
00004 #define PROGRAM_HH
00005
00006 #include <fstream>
00007 #include <iostream>
00008
00009 using namespace std;
00010
00022 class Program{
00023 protected:
00031   int rozmiar_tab;
00032
00039   int *tab;
00040
00047   ifstream plik_we;
00048
00055   ofstream plik_wy;
00056
00057 public:
00063   int getRozmiar_tab(){return rozmiar_tab;}
00064
```

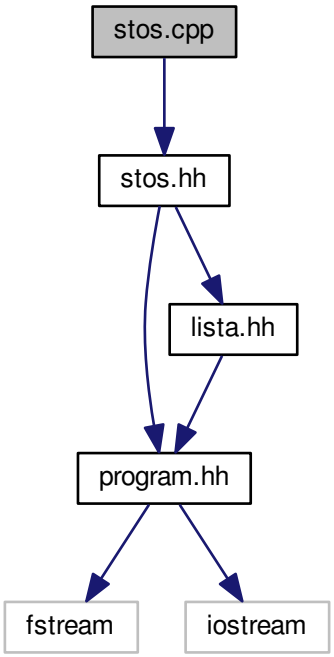
```
00071   Program(){rozmiar_tab=0;tab=NULL;}
00072
00079   ~Program(){delete[] tab; tab=NULL;}
00080
00096   bool wczytaj_dane(char* nazwa_pliku);
00097
00114   bool wczytaj_dane(char* nazwa_pliku, int ile_danych);
00115
00131  bool zapisz_dane(char* nazwa_pliku);
00132
00138   void wyswietl_dane();
00139
00145   virtual bool wykonaj_program();
00146
00147   virtual bool wykonaj_program(char* nazwa_pliku, int ilosc_danych)=0;
00148   virtual void wyczysc_dane(int ile)=0;
00149 };
00150
00151 #endif
```

5.23 Dokumentacja pliku stos.cpp

Zawiera definicje metod klasy Stos.

```
#include "stos.hh"
```

Wykres zależności załączania dla stos.cpp:



5.24 stos.cpp

```
00001 //stos.cpp
00002 #include "stos.hh"
00007 void Stos::push(int x){
00008   pole *nowe = new pole;
00009   nowe->wartosc = x;
```

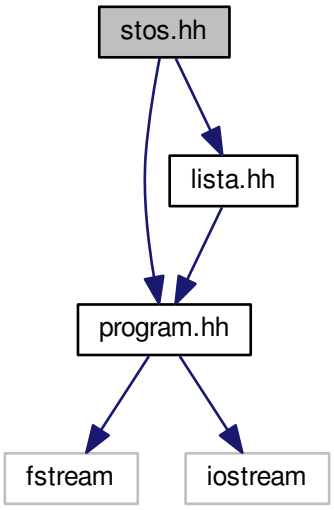
```
00010   nowe->next=top;
00011   top=nowe;
00012 }
00013
00014 void Stos::pop(){
00015   if(top == NULL)
00016     cerr<<"Stos jest pusty!"<<endl;
00017   else{
00018     pole *tmp = top;
00019     top=top->next;
00020     delete tmp;
00021   }
00022 }
00023
00024 int Stos::size(){
00025
00026   if(top == NULL)
00027     return 0;
00028   else{
00029     pole *wsk = top;
00030     int i=1;
00031     while(wsk->next){
00032       wsk=wsk->next;
00033       i++;
00034     }
00035   return i;
00036   }
00037 }
00038
00039 bool Stos::wykonaj_program(char* nazwa_pliku, int ilosc_danych){
00040   int tmp;
00041   plik_we.open(nazwa_pliku);
00042   if(plik_we.good()==false){
00043     cerr<<"Blad odczytu pliku!"<<endl;
00044     return false;
00045   }
00046   plik_we >> rozmiar_tab;
00047   for(int i=0;i<ilosc_danych;i++){
00048     plik_we >> tmp;
00049     push(tmp);
00050   }
00051   plik_we.close();
00052   return true;
00053 }
00054
00055 void Stos::wyczysc_dane(int ile){
00056   for(int i=0;i<ile;i++)
00057     pop();
00058 }
```

5.25 Dokumentacja pliku stos.hh

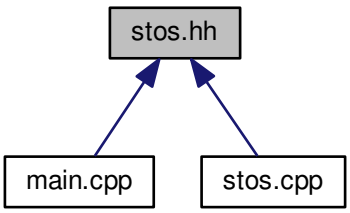
Definicja klasy Stos.

```
#include "program.hh"
#include "lista.hh"
```

Wykres zależności załączania dla stos.hh:



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



Komponenty

- class Stos

5.26 stos.hh

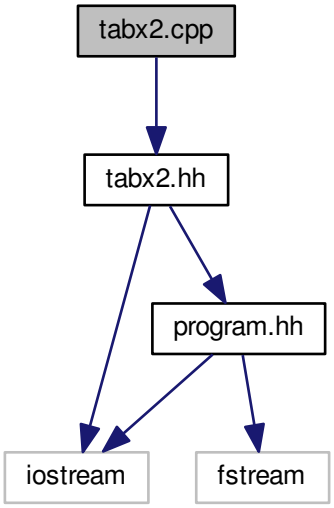
```
00001 //stos.hh
00002 #ifndef STOS_HH
00003 #define STOS_HH
00004 #include "program.hh"
00005 #include "lista.hh"
00006
00013 class Stos: public Program{
00014   pole *top;
00015 public:
```

```
00021   Stos(){
00022     top = NULL;
00023   }
00031   void push(int x);
00037   void pop();
00045   int size();
00046
00056   bool wykonaj_program(char* nazwa_pliku,int ilosc_danych);
00057
00065   void wyczysc_dane(int ile);
00066 };
00067
00068 #endif
```

5.27 Dokumentacja pliku tabx2.cpp

Plik zawiera metody klasy Tabx2.

`#include "tabx2.hh"`
Wykres zależności załączania dla tabx2.cpp:



5.28 tabx2.cpp

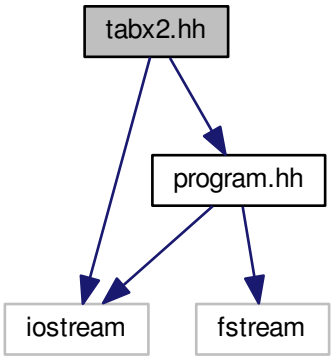
```
00001 #include "tabx2.hh"
00002
00003 using namespace std;
00007 bool Tabx2::wykonaj_program(){
00008   if(rozmiar_tab==0){
00009     cerr<<"Brak danych wejsciowych!"<<endl;
00010     return false;
00011   }
00012   for(int i=0;i<rozmiar_tab;i++){
00013     tab[i]*=2;
00014   }
00015   return true;
00016 }
```

5.29 Dokumentacja pliku tabx2.hh

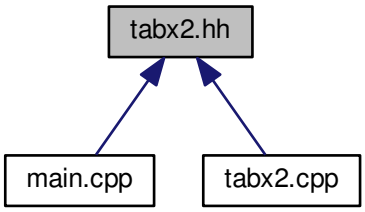
Definicja klasy Tabx2.

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

Wykres zależności załączania dla tabx2.hh:



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



Komponenty

- class Tabx2

5.29.1 Opis szczegółowy

Klasa Tabx2 jest klasa pochodna od klasy Program. Definiuje metode podawajaca kazda liczbe znajdujaca sie w tablicy danych wskazywanej przez zmienna tab klasy Program.

Definicja w pliku tabx2.hh.

5.30 tabx2.hh

```
00001 //mnozenie_tablicy.hh
00002
00003 #ifndef TABX2_HH
00004 #define TABX2_HH
00005
00015 #include <iostream>
00016 #include "program.hh"
00017
00018 class Tabx2: public Program{
00019
00020 public:
00030   virtual bool wykonaj_program();
00031 };
00032
00033 #endif
```

Skorowidz

~Lista_tab
    Lista_tab, 19
~Program
    Program, 24

Benchmark, 7
    czas_pomiaru, 10
    rozpocznij_pomiar, 7
    t1, 10
    t2, 10
    testuj, 8
    testuj_strukture, 8
    zakoncz_pomiar, 9
benchmark.cpp, 33, 34
benchmark.hh, 34, 35

czas_pomiaru
    Benchmark, 10

first
    Lista, 17

getRozmiar_tab
    Program, 24

head
    Kolejka, 13

iterator
    Lista_tab, 21

Kolejka, 10
    head, 13
    Kolejka, 11
    pop, 12
    push, 12
    size, 12
    tail, 13
    wyczysc_dane, 12
    wykonaj_program, 13
kolejka.cpp, 36, 37
kolejka.hh, 37, 38

LISTA__TAB_HH
    lista_tab.hh, 44
Lista, 14
    first, 17
    Lista, 15
    pop, 15
    push, 15
    size, 16
    wyczysc_dane, 16
    wykonaj_program, 16
lista.cpp, 39
lista.hh, 40, 41
Lista_tab, 17
    ~Lista_tab, 19
    iterator, 21
    Lista_tab, 19
    Lista_tab, 19
    pop, 19
    push, 19
    rozmiar, 21
    size, 19
    tab, 21
    wyczysc_dane, 19
    wykonaj_program, 21
lista_tab.cpp, 42
lista_tab.hh, 44, 45
    LISTA__TAB_HH, 44

main
    main.cpp, 46
main.cpp, 45, 46
    main, 46

next
    pole, 22

plik_we
    Program, 26
plik_wy
    Program, 26
pole, 21
    next, 22
    pole, 22
    wartosc, 22
pop
    Kolejka, 12
    Lista, 15
    Lista_tab, 19
    Stos, 28
Program, 22
    ~Program, 24
    getRozmiar_tab, 24
    plik_we, 26
    plik_wy, 26
    Program, 24
    rozmiar_tab, 26
    tab, 26
    wczytaj_dane, 24

wyczysc_dane, 25
wykonaj_program, 25
wyswietl_dane, 25
zapisz_dane, 25
program.cpp, 47
program.hh, 48, 49
push
Kolejka, 12
Lista, 15
Lista_tab, 19
Stos, 28

rozmiar
Lista_tab, 21
rozmiar_tab
Program, 26
rozpocznij_pomiar
Benchmark, 7

size
Kolejka, 12
Lista, 16
Lista_tab, 19
Stos, 28
Stos, 26
pop, 28
push, 28
size, 28
Stos, 28
top, 30
wyczysc_dane, 29
wykonaj_program, 29
stos.cpp, 50
stos.hh, 51, 52

t1
Benchmark, 10
t2
Benchmark, 10
tab
Lista_tab, 21
Program, 26
Tabx2, 30
wykonaj_program, 31
tabx2.cpp, 53
tabx2.hh, 54, 55
tail
Kolejka, 13
testuj
Benchmark, 8
testuj_strukture
Benchmark, 8
top
Stos, 30

wartosc
pole, 22
wczytaj_dane
Program, 24
wyczysc_dane
Kolejka, 12
Lista, 16
Lista_tab, 19
Program, 25
Stos, 29
wykonaj_program
Kolejka, 13
Lista, 16
Lista_tab, 21
Program, 25
Stos, 29
Tabx2, 31
wyswietl_dane
Program, 25

zakoncz_pomiar
Benchmark, 9
zapisz_dane
Program, 25