

Temat 5A

Wygenerowano przez Doxygen 1.8.20

1 README	1
2 Indeks hierarchiczny	3
2.1 Hierarchia klas	3
3 Indeks klas	5
3.1 Lista klas	5
4 Indeks plików	7
4.1 Lista plików	7
5 Dokumentacja klas	9
5.1 Dokumentacja szablonu klasy Interfejs< T >	9
5.1.1 Dokumentacja konstruktora i destruktora	10
5.1.1.1 Interfejs()	11
5.1.1.2 ~Interfejs()	12
5.1.2 Dokumentacja funkcji składowych	12
5.1.2.1 stworzZbiory()	12
5.1.2.2 wprowadzDane()	12
5.1.2.3 wykonajDzialanie()	13
5.1.2.4 wyswietlOpcje()	13
5.2 Dokumentacja szablonu klasy Lista< T >	13
5.2.1 Dokumentacja konstruktora i destruktora	14
5.2.1.1 Lista() [1/2]	15
5.2.1.2 Lista() [2/2]	15
5.2.1.3 ~Lista()	15
5.2.2 Dokumentacja funkcji składowych	15
5.2.2.1 czyNalezyDoListy()	15
5.2.2.2 dodajDoListy()	15
5.2.2.3 ileElementowLista()	16
5.2.2.4 oproznijListe()	16
5.2.2.5 usunZListy()	16
5.2.2.6 wyswietlListe()	17
5.2.3 Dokumentacja atrybutów składowych	17
5.2.3.1 pHead	17
5.3 Dokumentacja szablonu klasy Zbior< T >	17
5.3.1 Dokumentacja konstruktora i destruktora	19
5.3.1.1 Zbior() [1/2]	19
5.3.1.2 Zbior() [2/2]	19
5.3.1.3 ~Zbior()	19
5.3.2 Dokumentacja funkcji składowych	19
5.3.2.1 czyNalezy()	19
5.3.2.2 dodaj()	20
5.3.2.3 ileElementow()	20

5.3.2.4 operator*()	20
5.3.2.5 operator*=(())	21
5.3.2.6 operator+()	21
5.3.2.7 operator+=(())	21
5.3.2.8 operator-()	22
5.3.2.9 operator-=(())	22
5.3.2.10 oproznij()	22
5.3.2.11 usun()	23
5.3.2.12 wyswietl()	23
6 Dokumentacja plików	25
6.1 Dokumentacja pliku CMakeLists.txt	25
6.2 Dokumentacja pliku interfejs.h	25
6.3 Dokumentacja pliku lista.h	26
6.4 Dokumentacja pliku main.cpp	26
6.4.1 Dokumentacja funkcji	27
6.4.1.1 main()	27
6.5 Dokumentacja pliku README.md	28
6.6 Dokumentacja pliku zbior.h	28
6.6.1 Dokumentacja funkcji	29
6.6.1.1 operator<<()	29
6.6.1.2 operator>>()	29
Indeks	31

Rozdział 1

README

Tutaj umieszczać projekt

Rozdział 2

Indeks hierarchiczny

2.1 Hierarchia klas

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

Lista< T >	13
Zbior< T >	17
Interfejs< T >	9

Rozdział 3

Indeks klas

3.1 Lista klas

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

Interfejs< T >	9
Lista< T >	13
Zbior< T >	17

Rozdział 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

interfejs.h	25
lista.h	26
main.cpp	26
zbior.h	28

Rozdział 5

Dokumentacja klas

5.1 Dokumentacja szablonu klasy Interfejs< T >

```
#include <interfejs.h>
```

Diagram dziedziczenia dla Interfejs< T >

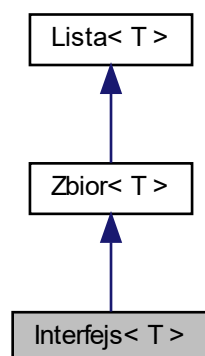
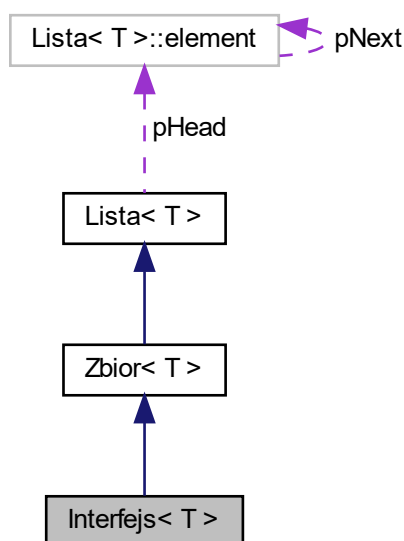


Diagram współpracy dla Interfejs< T >:



Metody publiczne

- [Interfejs](#) (const int &ileZbiorow, const int &flaga)
- [~Interfejs](#) ()=default
- void [stworzZbiory](#) (const int &flaga)

Metoda wywołuje w petli [Interfejs::wprowadzDane](#), dokładnie tyle razy ile tego wymaga użytkownik.

- void [wprowadzDane](#) (const int &flaga)

Metoda tworzy nowy obiekt typu [Zbior](#), odczytuje znaki wprowadzane w terminalu i wpisuje je do listy. Na końcu wprowadza wskaźnik nowo utworzonego zbioru do vectora.

- void [wyswietlOpcje](#) ()

Metoda wyświetla możliwe operacje na zbiorach, oczekuje na wprowadzenie przez użytkownika cyfry odpowiadającej operacji, oraz wywołuje metodę [Interfejs::wykonajDzialanie](#) z odpowiednim parametrem.

- void [wykonajDzialanie](#) (const int &flaga)

Metoda oczekuje na wprowadzenie przez użytkownika cyfr / cyfry odpowiadającej wybranemu zbiorowi, pobiera wskaźnik wybranego zbioru z vectora, wywołuje odpowiednią metodę operacji arytmetycznej, na końcu wypisuje nowo utworzony zbior.

Dodatkowe Dziedziczone Składowe

5.1.1 Dokumentacja konstruktora i destruktor

5.1.1.1 Interfejs()

```
template<class T >
Interfejs< T >::Interfejs (
    const int & ileZbiorow,
    const int & flaga )
```

Parametry

<i>ileZbiorow</i>	- ilosc zbiorow do utworzenia
<i>flaga</i>	- okresla typ zbioru, 1 - int, 2 - double, 3 - char

5.1.1.2 ~Interfejs()

```
template<class T >
Interfejs< T >::~~Interfejs ( ) [default]
```

5.1.2 Dokumentacja funkcji składowych

5.1.2.1 stworzZbiory()

```
template<class T >
void Interfejs< T >::stworzZbiory (
    const int & flaga )
```

Metoda wywołuje w petli [Interfejs::wprowadzDane](#), dokładnie tyle razy ile tego wymaga użytkownik.

Parametry

<i>flaga</i>	- okresla typ zbioru, 1 - int, 2 - double, 3 - char
--------------	---

5.1.2.2 wprowadzDane()

```
template<class T >
void Interfejs< T >::wprowadzDane (
    const int & flaga )
```

Metoda tworzy nowy obiekt typu [Zbior](#), odczytuje znaki wprowadzane w terminalu i wpisuje je do listy. Na koncu wprowadza wskaźnik nowo utworzonego zbioru do vectora.

Parametry

<i>flaga</i>	- okresla typ zbioru, 1 - int, 2 - double, 3 - char
--------------	---

5.1.2.3 wykonajDzialanie()

```
template<class T >
void Interfejs< T >::wykonajDzialanie (
    const int & flaga )
```

Metoda oczekuje na wprowadzenie przez uzytkownika cyfr / cyfry odpowiadajacej wybranemu zbiorowi, pobiera wskaznik wybranego zbioru z vectora, wywoluje odpowiednia metode operacji arytmetycznej, na koncu wypisuje nowo utworzony zbior.

Parametry

<i>flaga</i>	- okresla typ zbioru, 1 - int, 2 - double, 3 - char
--------------	---

5.1.2.4 wyswietlOpcje()

```
template<class T >
void Interfejs< T >::wyswietlOpcje
```

Metoda wyswietla mozliwe operacje na zbiorach, oczekuje na wprowadzenie przez uzytkownika cyfry odpowiadajacej operacji, oraz wywoluje metode [Interfejs::wykonajDzialanie](#) z odpowiednim parametrem.

Dokumentacja dla tej klasy zostala wygenerowana z pliku:

- [interfejs.h](#)

5.2 Dokumentacja szablonu klasy Lista< T >

```
#include <lista.h>
```

Diagram dziedziczenia dla Lista< T >

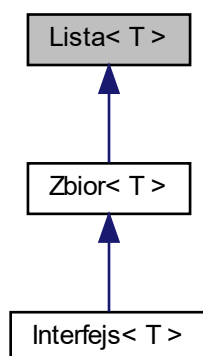
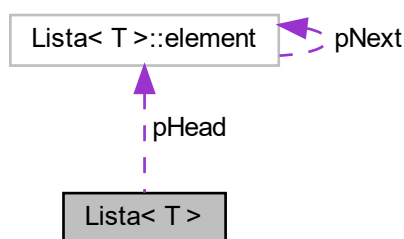


Diagram współpracy dla `Lista< T >`:



Metody publiczne

- `Lista()`
Konstruktor z domyślną inicjalizacją `pHead(nullptr)`
- `Lista(const Lista< T > &_lista)`
Konstruktor kopiujący dla klasy `Lista`.
- `~Lista()`

Metody chronione

- `void usunZListy(const T &value)`
Metoda przekazuje zadaną wartość do usunięcia, następnie w metodzie sprawdzane jest czy element do usunięcia znajduje się na początku listy czy nie. Wykonywana jest operacja delete, a na końcu przywracane są adresy początkowe głowy listy.
- `void oproznijListe()`
Metoda usuwa całą listę jednokierunkową oraz zwalnia zaalokowaną wcześniej pamięć.
- `void wyswietlListe(ostream &os)`
Metoda wyświetla zbiór, który jest przechowywany w liście jednokierunkowej. Po zakończeniu wyświetlania wskaźniki ustawiane są na adresy początkowe.
- `void dodajDoListy(const T &value)`
Metoda dodaje wartość do listy jednokierunkowej.
- `int ileElementowLista()`
Metoda ma za zadanie przejście całej listy jednokierunkowej w celu zliczenia ilości elementów w zbiorze.
- `bool czyNależyDoListy(const T &value)`

Atrybuty chronione

- `element * pHead`
Wskaźnik typu `element` na głowę listy.

5.2.1 Dokumentacja konstruktora i destruktoru

5.2.1.1 Lista() [1/2]

```
template<class T >
Lista< T >::Lista ( ) [inline]
```

Konstruktor z domyslna inicjalizacja pHead(nullptr)

5.2.1.2 Lista() [2/2]

```
template<class T >
Lista< T >::Lista (
    const Lista< T > & _lista )
```

Konstruktor kopiujacy dla klasy [Lista](#).

5.2.1.3 ~Lista()

```
template<class T >
Lista< T >::~~Lista
```

5.2 Dokumentacja funkcji składowych

5.2.2.1 czyNalezyDoListy()

```
template<class T >
bool Lista< T >::czyNalezyDoListy (
    const T & value ) [protected]
```

Parametry

<i>value</i>	- wartosc, ktora jest przekazywana do sprawdzenia czy nalezy do zbioru
--------------	--

Zwraca

Zwraca true jezeli wartosc nalezy do zbioru, false w przeciwnym wypadku

5.2.2.2 dodajDoListy()

```
template<class T >
void Lista< T >::dodajDoListy (
    const T & value ) [protected]
```

Metoda dodaje wartosc do listy jednokierunkowej.

Parametry

<i>value</i>	- wartosc, ktora nalezy wpisac do zbioru
--------------	--

5.2.2.3 ileElementowLista()

```
template<class T >
int Lista< T >::ileElementowLista [protected]
```

Metoda ma za zadanie przejrzenie całej listy jednokierunkowej w celu zliczenia ilości elementów w zbiorze.

Zwraca

Zwraca ilość elementów w zbiorze

5.2.2.4 oproznijListe()

```
template<class T >
void Lista< T >::opoznijListe [protected]
```

Metoda usuwa całą listę jednokierunkową oraz zwalnia zaalokowaną wcześniej pamięć.

5.2.2.5 usunZListy()

```
template<class T >
void Lista< T >::usunZListy (
    const T & value ) [protected]
```

Metoda przekazuje zadana wartość do usunięcia, następnie w metodzie sprawdzane jest czy element do usunięcia znajduje się na początku listy czy nie. Wykonywana jest operacja delete, a na końcu przywracane są adresy początkowe głowy listy.

Parametry

<i>value</i>	- wartość do usunięcia ze zbioru
--------------	----------------------------------

5.2.2.6 wyswietlListe()

```
template<class T >
void Lista< T >::wyswietlListe (
    ostream & os ) [protected]
```

Metoda wyswietla zbior, ktory jest przechowywany w liscie jednokierunkowej. Po zakonczeniu wyswietlania wskaźniki ustawiane sa na adresy początkowe.

Parametry

<i>ostream</i>	&os - strumien wyjsciowy
----------------	--------------------------

5.2.3 Dokumentacja atrybutów składowych

5.2.3.1 pHead

```
template<class T >
element* Lista< T >::pHead [protected]
```

Wskaźnik typu element na glowe listy.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [lista.h](#)

5.3 Dokumentacja szablonu klasy Zbior< T >

```
#include <zbior.h>
```

Diagram dziedziczenia dla Zbior< T >

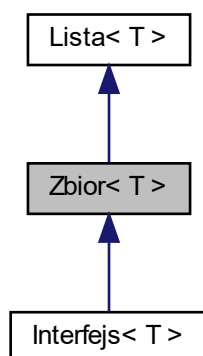
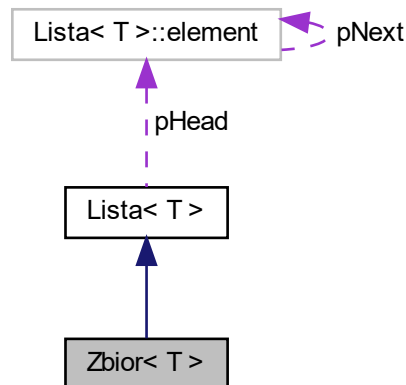


Diagram współpracy dla `Zbior< T >`:



Metody publiczne

- `Zbior()`=default
- `Zbior(const Zbior< T > &_zbior)`
- `~Zbior()`=default
- void `usun(const T &value)`
Metoda wywołuje metodę `Lista::usunZListy`, odziedziczona z klasy `Lista`.
- void `opoznij()`
Metoda wywołuje metodę `Lista::opoznijListe`, odziedziczona z klasy `Lista`.
- void `wyswietl(ostream &os)`
Metoda wywołuje metodę `Lista::wyswietlListe`, odziedziczona z klasy `Lista`.
- void `dodaj(const T &value)`
Metoda wywołuje metodę `Lista::dodajDoListy`, odziedziczona z klasy `Lista`.
- int `ileElementow()`
Metoda wywołuje metodę `Lista::ileElementowListe`, odziedziczona z klasy `Lista`.
- bool `czyNalezy(const T &value)`
Metoda wywołuje metodę `Lista::czyNalezyDoListy`, odziedziczona z klasy `Lista`.
- `Zbior< T > operator+(Zbior< T > &_zbior)`
Przeciążony operator sumy.
- `Zbior< T > operator-(Zbior< T > &_zbior)`
Przeciążony operator różnicy.
- `Zbior< T > operator*(Zbior< T > &_zbior)`
- `Zbior operator+=(Zbior< T > &_zbior)`
Metoda zaimplementowana inline, wykorzystuje `operator+` do którego przekazuje wybrany zbior.
- `Zbior operator-=(Zbior< T > &_zbior)`
Metoda zaimplementowana inline, wykorzystuje `operator-` do którego przekazuje wybrany zbior.
- `Zbior operator*=(Zbior< T > &_zbior)`
Metoda zaimplementowana inline, wykorzystuje `operator*` do którego przekazuje wybrany zbior.

Dodatkowe Dziedziczone Składowe

5.3.1 Dokumentacja konstruktora i destruktora

5.3.1.1 Zbior() [1/2]

```
template<class T >
Zbior< T >::Zbior ( ) [default]
```

5.3.1.2 Zbior() [2/2]

```
template<class T >
Zbior< T >::Zbior (
    const Zbior< T > & _zbior )
```

5.3.1.3 ~Zbior()

```
template<class T >
Zbior< T >::~~Zbior ( ) [default]
```

5.3.2 Dokumentacja funkcji składowych

5.3.2.1 czyNalezy()

```
template<class T >
bool Zbior< T >::czyNalezy (
    const T & value )
```

Metoda wywołuje metodę [Lista::czyNalezyDoListy](#), odziedziczona z klasy [Lista](#).

Parametry

<i>value</i>	- wartosc, ktora jest przekazywana do sprawdzenia czy nalezy do zbioru
--------------	--

Zwraca

Zwraca true jezeli wartosc nalezy do zbioru, false w przeciwnym wypadku

5.3.2.2 dodaj()

```
template<class T >
void Zbior< T >::dodaj (
    const T & value )
```

Metoda wywołuje metode [Lista::dodajDoListy](#), odziedziczona z klasy [Lista](#).

Parametry

<i>value</i>	- wartosc, ktora nalezy wpisac do zbioru
--------------	--

5.3.2.3 ileElementow()

```
template<class T >
int Zbior< T >::ileElementow
```

Metoda wywołuje metode [Lista::ileElementowLista](#), odziedziczona z klasy [Lista](#).

Zwraca

Zwraca ilosc elementow w zbiorze

5.3.2.4 operator*()

```
template<typename T >
Zbior< T > Zbior< T >::operator* (
    Zbior< T > & _zbior )
```

Parametry

<i>_zbior</i>	adres drugiego zbioru
---------------	-----------------------

Zwraca

Zwraca nowy zbior bedacy wynikiem dzialania arytmetycznego iloczynu zbiorow

5.3.2.5 operator*=()

```
template<typename T >
Zbior< T > Zbior< T >::operator*= (
    Zbior< T > & _zbior ) [inline]
```

Metoda zaimplementowana inline, wykorzystuje operator* do którego przekazuje wybrany zbior.

Parametry

<code>_zbior</code>	adres drugiego zbioru
---------------------	-----------------------

Zwraca

Zwraca nowy zbior bedacy wynikiem dzialania arytmetycznego iloczynu zbiorow

5.3.2.6 operator+()

```
template<typename T >
Zbior< T > Zbior< T >::operator+ (
    Zbior< T > & _zbior )
```

Przeciazony operator sumy.

Parametry

<code>_zbior</code>	- adres drugiego zbioru
---------------------	-------------------------

Zwraca

Zwraca nowy zbior bedacy wynikiem dzialania arytmetycznego sumy zbiorow

5.3.2.7 operator+=()

```
template<typename T >
Zbior< T > Zbior< T >::operator+= (
    Zbior< T > & _zbior ) [inline]
```

Metoda zaimplementowana inline, wykorzystuje operator+ do którego przekazuje wybrany zbior.

Parametry

<code>_zbior</code>	adres drugiego zbioru
---------------------	-----------------------

Zwraca

Zwraca nowy zbior bedacy wynikiem dzialania arytmetycznego sumy zbiorow

5.3.2.8 operator-()

```
template<typename T >
Zbior< T > Zbior< T >::operator- (
    Zbior< T > & _zbior )
```

Przeciazony operator roznicy.

Parametry

<code>_zbior</code>	adres drugiego zbioru
---------------------	-----------------------

Zwraca

Zwraca nowy zbior bedacy wynikiem dzialania arytmetycznego roznicy zbiorow

5.3.2.9 operator-=()

```
template<typename T >
Zbior< T > Zbior< T >::operator-= (
    Zbior< T > & _zbior ) [inline]
```

Metoda zaimplementowana inline, wykorzystuje operator- do ktorego przekazuje wybrany zbior.

Parametry

<code>_zbior</code>	adres drugiego zbioru
---------------------	-----------------------

Zwraca

Zwraca nowy zbior bedacy wynikiem dzialania arytmetycznego roznicy zbiorow

5.3.2.10 oproznij()

```
template<class T >
void Zbior< T >::oproznij
```

Metoda wywoluje metode [Lista::oproznijListe](#), odziedziczona z klasy [Lista](#).

5.3.2.11 usun()

```
template<class T >
void Zbior< T >::usun (
    const T & value )
```

Metoda wywołuje metode [Lista::usunZListy](#), odziedziczona z klasy [Lista](#).

Parametry

<i>value</i>	- value - wartosc do usuniecia ze zbioru
--------------	--

5.3.2.12 wyswietl()

```
template<class T >
void Zbior< T >::wyswietl (
    ostream & os )
```

Metoda wywołuje metode [Lista::wyswietlListe](#), odziedziczona z klasy [Lista](#).

Parametry

<i>os</i>	- strumien wyjsciowy
-----------	----------------------

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [zbior.h](#)

Rozdział 6

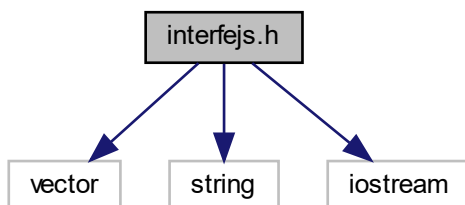
Dokumentacja plików

6.1 Dokumentacja pliku CMakeLists.txt

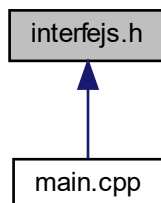
6.2 Dokumentacja pliku interfejs.h

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

Wykres zależności załączania dla interfejs.h:



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



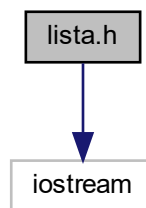
Komponenty

- class `Interfejs< T >`

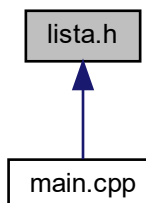
6.3 Dokumentacja pliku lista.h

```
#include <iostream>
```

Wykres zależności załączania dla lista.h:



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



Komponenty

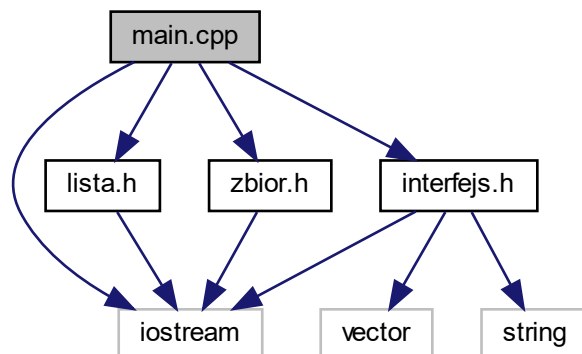
- class `Lista< T >`

6.4 Dokumentacja pliku main.cpp

```
#include <iostream>
#include "lista.h"
#include "zbior.h"
```

```
#include "interfejs.h"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- `int main ()`

6.4.1 Dokumentacja funkcji

6.4.1.1 main()

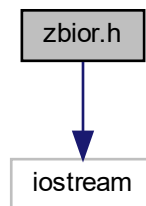
```
int main ( )
```

6.5 Dokumentacja pliku README.md

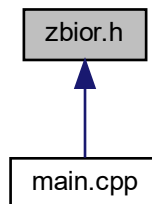
6.6 Dokumentacja pliku zbior.h

```
#include <iostream>
```

Wykres zależności załączania dla zbior.h:



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



Komponenty

- class `Zbior< T >`

Funkcje

- `template<class T >`
`ostream & operator<< (ostream &osOutput, Zbior< T > &zbior)`
- `template<class T >`
`istream & operator>> (istream &isInput, Zbior< T > &zbior)`

6.6.1 Dokumentacja funkcji

6.6.1.1 operator<<()

```
template<class T >
ostream & operator<< (
    ostream & osOutput,
    Zbior< T > & zbior )
```

6.6.1.2 operator>>()

```
template<class T >
istream & operator>> (
    istream & isInput,
    Zbior< T > & zbior )
```


Indeks

~Interfejs
Interfejs< T >, 12

~Lista
Lista< T >, 15

~Zbior
Zbior< T >, 19

CMakeLists.txt, 25

czyNalezy
Zbior< T >, 19

czyNalezyDoListy
Lista< T >, 15

dodaj
Zbior< T >, 20

dodajDoListy
Lista< T >, 15

ileElementow
Zbior< T >, 20

ileElementowLista
Lista< T >, 16

Interfejs
Interfejs< T >, 10

Interfejs< T >, 9
~Interfejs, 12
Interfejs, 10
stworzZbiory, 12
wprowadzDane, 12
wykonajDzialanie, 12
wyswietlOpcje, 13

interfejs.h, 25

Lista
Lista< T >, 14, 15

Lista< T >, 13
~Lista, 15
czyNalezyDoListy, 15
dodajDoListy, 15
ileElementowLista, 16
Lista, 14, 15
oproznijListe, 16
pHead, 17
usunZListy, 16
wyswietlListe, 16

lista.h, 26

main
main.cpp, 27

main.cpp, 26
main, 27

operator<<
zbior.h, 29

operator>>
zbior.h, 29

operator*
Zbior< T >, 20

operator*=
Zbior< T >, 20

operator+
Zbior< T >, 21

operator+=
Zbior< T >, 21

operator-
Zbior< T >, 22

operator-=
Zbior< T >, 22

oproznij
Zbior< T >, 22

oproznijListe
Lista< T >, 16

pHead
Lista< T >, 17

README.md, 28

stworzZbiory
Interfejs< T >, 12

usun
Zbior< T >, 22

usunZListy
Lista< T >, 16

wprowadzDane
Interfejs< T >, 12

wykonajDzialanie
Interfejs< T >, 12

wyswietl
Zbior< T >, 23

wyswietlListe
Lista< T >, 16

wyswietlOpcje
Interfejs< T >, 13

Zbior
Zbior< T >, 19

Zbior< T >, 17
~Zbior, 19
czyNalezy, 19
dodaj, 20

- ileElementow, [20](#)
- operator*, [20](#)
- operator*=[, 20](#)
- operator+, [21](#)
- operator+=, [21](#)
- operator-, [22](#)
- operator-=, [22](#)
- opoznij, [22](#)
- usun, [22](#)
- wyswietl, [23](#)
- Zbior, [19](#)
- zbior.h, [28](#)
 - operator<<, [29](#)
 - operator>>, [29](#)