# Temat 5A

Wygenerowano przez Doxygen 1.8.20

# 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:

# Rozdział 3

# Indeks klas

## 3.1 Lista klas

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

# Rozdział 4

# Indeks plików

## 4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

# Rozdział 5

# Dokumentacja klas

## 5.1 Dokumentacja szablonu klasy Interfejs< T >

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

Diagram dziedziczenia dla Interfejs< T >

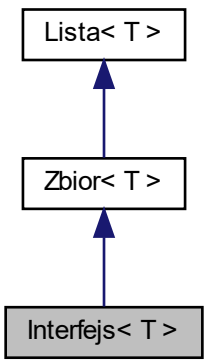
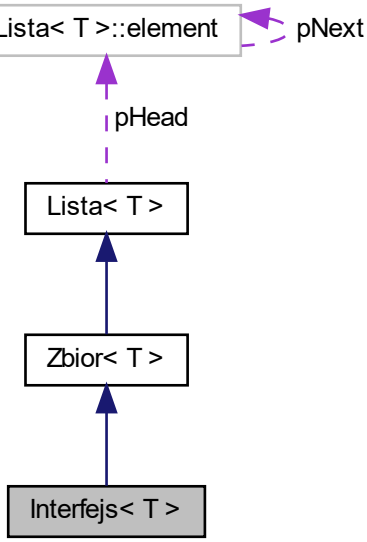


Diagram współpracy dla Interfejs< T >:




## Metody publiczne

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

  *Metoda wywoluje w petli Interfejs::wprowadzDane, dokladnie tyle razy ile tego wymaga uzytkownik.*
- void wprowadzDane (const int &flaga)

  *Metoda tworzy nowy obiekt typu Zbior, odczytuje znaki wprowadzane w terminalu i wpisuje je do listy. Na koncu wprowadza wskaznik nowo utworzonego zbioru do vectora.*
- void wyswietlOpcje ()

  *Metoda wyswietla mozliwe operacjie na zbiorach, oczekuje na wprowadzenie przez uzytkownika cyfry odpowiadajacej operacji, oraz wywoluje metode Interfejs::wykonajDzialanie z odpowiednim parametrem.*
- void 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.*

## Dodatkowe Dziedziczone Składowe

### 5.1.1 Dokumentacja konstruktora i destruktora

#### 5.1.1.1 Interfejs()

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

**Parametry**

| | |
|---|---|
| *ileZbiorow* | - ilosc zbiorow do utworzenia |
| *flaga* | - 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 wywoluje w petli Interfejs::wprowadzDane, dokladnie tyle razy ile tego wymaga uzytkownik.

**Parametry**

| | |
|---|---|
| *flaga* | - 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 wskaznik nowo utworzonego zbioru do vectora.

**Parametry**

| | |
|---|---|
| *flaga* | - 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**

| | |
|---|---|
| *flaga* | - okresla typ zbioru, 1 - int, 2 - double, 3 - char |

#### 5.1.2.4 wyswietlOpcje()

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

Metoda wyswietla mozliwe operacjie na zbiorach, oczekuje na wprowadzenie przez uzytkownika cyfry odpowiadajacej operacji, oraz wywoluje metode Interfejs::wykonajDzialanie z odpowiednim parametrem.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- interfejs.h

## 5.2 Dokumentacja szablonu klasy Lista< T >

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

Diagram dziedziczenia dla Lista< T >

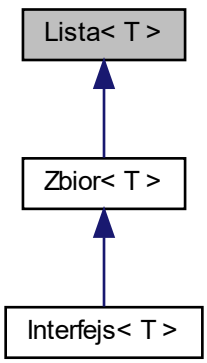
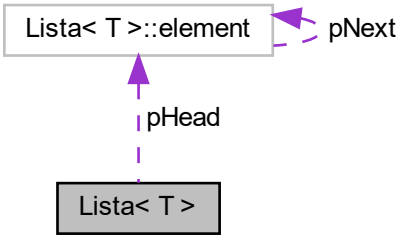


Diagram współpracy dla Lista< T >:




## Metody publiczne

- Lista ()

  *Konstruktor z domyslna inicjalizacja pHead(nullptr)*
- Lista (const Lista< T > &_lista)

  *Konstruktor kopiujacy dla klasy Lista.*
- ~Lista ()

## Metody chronione

- void usunZListy (const T &value)

  *Metoda przekazuje zadana wartosc do usuniecia, nastepnie w metodzie sprawdzane jest czy element do usuniecia znajduje sie na poczatku listy czy nie. Wykonywana jest operacja delete, a na koncu przywracane sa adresy poczatkowe glowy listy.*
- void oproznijListe ()

  *Metoda usuwa cala liste jednokierunkowa oraz zwalnia zaalokowana wczesniej pamiec.*
- void wyswietlListe (ostream &os)

  *Metoda wyswietla zbior, ktory jest przechowywany w liscie jednokierunkowej. Po zakonczeniu wyswietlania wskazniki ustawiane sa na adresy poczatkowe.*
- void dodajDoListy (const T &value)

  *Metoda dodaje wartosc do listy jednokierunkowej.*
- int ileElementowLista ()

  *Metoda ma za zadanie przejrzenie calej listy jendokierunkowej w celu zliczenia ilosci elementow w zbiorze.*
- bool czyNalezyDoListy (const T &value)

## Atrybuty chronione

- element * pHead

  *Wskaznik typu element na glowe listy.*

### 5.2.1 Dokumentacja konstruktora i destruktora

#### 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
```

### 5.2.2 Dokumentacja funkcji składowych

#### 5.2.2.1 czyNalezyDoListy()

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

**Parametry**

| | |
|---|---|
| *value* | - 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**

| | |
|---|---|
| *value* | - wartosc, ktora nalezy wpisac do zbioru |

#### 5.2.2.3 ileElementowLista()

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

Metoda ma za zadanie przejrzenie calej listy jendokierunkowej w celu zliczenia ilosci elementow w zbiorze.

**Zwraca**

Zwraca ilosc elementow w zbiorze

#### 5.2.2.4 oproznijListe()

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

Metoda usuwa cala liste jednokierunkowa oraz zwalnia zaalokowana wczesniej pamiec.

#### 5.2.2.5 usunZListy()

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

Metoda przekazuje zadana wartosc do usuniecia, nastepnie w metodzie sprawdzane jest czy element do usuniecia znajduje sie na poczatku listy czy nie. Wykonywana jest operacja delete, a na koncu przywracane sa adresy poczatkowe glowy listy.

**Parametry**

| | |
|---|---|
| *value* | - wartosc do usunieca 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 wskazniki ustawiane sa na adresy poczatkowe.

**Parametry**

| | |
|---|---|
| *ostream* | &os - strumien wyjsciowy |

### 5.2.3 Dokumentacja atrybutów składowych

#### 5.2.3.1 pHead

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

Wskaznik 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 >

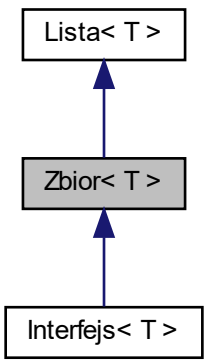
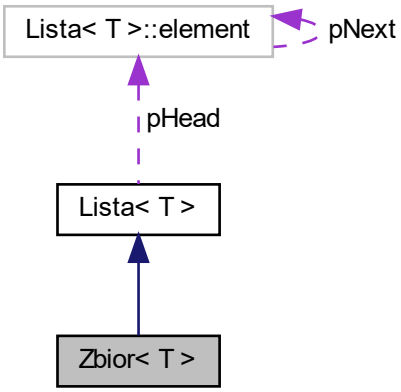


Diagram współpracy dla Zbior< T >:




## Metody publiczne

- Zbior ()=default
- Zbior (const Zbior< T > &_zbior)
- ~Zbior ()=default
- void usun (const T &value)

  *Metoda wywoluje metode Lista::usunZListy, odziedziczona z klasy Lista.*
- void oproznij ()

  *Metoda wywoluje metode Lista::oproznijListe, odziedziczona z klasy Lista.*
- void wyswietl (ostream &os)

  *Metoda wywoluje metode Lista::wyswietlListe, odziedziczona z klasy Lista.*
- void dodaj (const T &value)

  *Metoda wywoluje metode Lista::dodajDoListy, odziedziczona z klasy Lista.*
- int ileElementow ()

  *Metoda wywoluje metode Lista::ileElementowLista, odziedziczona z klasy Lista.*
- bool czyNalezy (const T &value)

  *Metoda wywoluje metode Lista::czyNalezyDoListy, odziedziczona z klasy Lista.*
- Zbior< T > operator+ (Zbior< T > &_zbior)

  *Przeciazony operator sumy.*
- Zbior< T > operator- (Zbior< T > &_zbior)

  *Przeciazony operator roznicy.*
- Zbior< T > operator* (Zbior< T > &_zbior)
- Zbior operator+= (Zbior< T > &_zbior)

  *Metoda zaimplementowana inline, wykorzystuje operator+ do ktorego przekazuje wybrany zbior.*
- Zbior operator-= (Zbior< T > &_zbior)

  *Metoda zaimplementowana inline, wykorzystuje operator- do ktorego przekazuje wybrany zbior.*
- Zbior operator*= (Zbior< T > &_zbior)

  *Metoda zaimplementowana inline, wykorzystuje operator* do ktorego 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 wywoluje metode Lista::czyNalezyDoListy, odziedziczona z klasy Lista.

**Parametry**

| | |
|---|---|
| *value* | - 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 wywoluje metode Lista::dodajDoListy, odziedziczona z klasy Lista.

**Parametry**

| | |
|---|---|
| *value* | - wartosc, ktora nalezy wpisac do zbioru |

#### 5.3.2.3 ileElementow()

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

Metoda wywoluje 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**

| | |
|---|---|
| *_zbior* | 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 ktorego przekazuje wybrany zbior.

**Parametry**

| | |
|---|---|
| *_zbior* | 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**

| | |
|---|---|
| *_zbior* | - 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 ktorego przekazuje wybrany zbior.

**Parametry**

| | |
|---|---|
| *_zbior* | 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**

| | |
|---|---|
| *_zbior* | 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**

| | |
|---|---|
| *_zbior* | 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 wywoluje metode Lista::usunZListy, odziedziczona z klasy Lista.

**Parametry**

| | |
|---|---|
| *value* | - value - wartosc do usunieca ze zbioru |

#### 5.3.2.12 wyswietl()

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

Metoda wywoluje metode Lista::wyswietlListe, odziedziczona z klasy Lista.

**Parametry**

| | |
|---|---|
| *os* | - strumien wyjsciowy |

Dokumentacja dla tej klasy została wygenerowana z pliku:

- zbior.h

# Rozdział 6

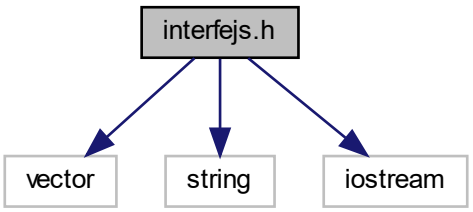
# Dokumentacja plików

## 6.1 Dokumentacja pliku CMakeLists.txt

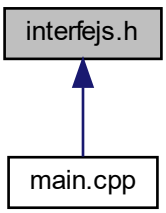
## 6.2 Dokumentacja pliku interfejs.h

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

Wykres zależności załączania dla interfejs.h:




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



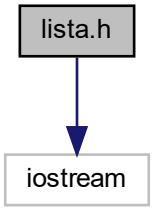
### Komponenty

- class Interfejs< T >

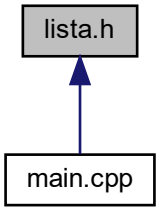
## 6.3 Dokumentacja pliku lista.h

```
#include <iostream>
```

Wykres zależności załączania dla lista.h:




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




### Komponenty

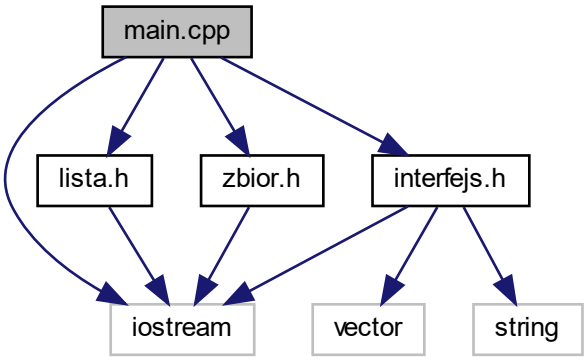
- class Lista< T >

## 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

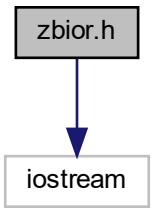
#### 6.4.1.1 main()

```
int main ( )
```

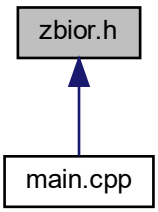
## 6.5 Dokumentacja pliku README.md

## 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