

## My Project

Generated by Doxygen 1.8.16



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 Drzewo Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 current_dist	5
3.1.2.2 Miasto	6
3.1.2.3 pLewy	6
3.1.2.4 pOdleglosci	6
3.1.2.5 poprzednik_w_najkrotszej_trasie	6
3.1.2.6 pPrawy	6
3.2 Odleglosc Struct Reference	6
3.2.1 Detailed Description	7
3.2.2 Member Data Documentation	7
3.2.2.1 kilometry	7
3.2.2.2 pNastepny	7
3.2.2.3 Polaczenie	7
3.3 Parametry Struct Reference	7
3.3.1 Detailed Description	8
3.3.2 Member Data Documentation	8
3.3.2.1 plikwejscowy	8
3.3.2.2 plikwyjscowy	8
3.3.2.3 siedziba	8
3.4 PorownajDrzewa Struct Reference	8
3.4.1 Detailed Description	9
3.4.2 Member Function Documentation	9
3.4.2.1 operator>()	9
3.5 Trasa Struct Reference	9
3.5.1 Detailed Description	9
3.5.2 Member Data Documentation	9
3.5.2.1 Miasto	10
3.5.2.2 Nastepny	10
<b>4 File Documentation</b>	<b>11</b>
4.1 funkcje.cpp File Reference	11
4.1.1 Function Documentation	11
4.1.1.1 AlgorytmDijkstry()	11
4.1.1.2 DodajdoDrzewa()	12

4.1.1.3 DodawanieDoListy()	12
4.1.1.4 odczytajParametry()	12
4.1.1.5 Usun()	14
4.1.1.6 UsunListy()	14
4.1.1.7 UsunStruktureTrasy()	14
4.1.1.8 WczytajDane()	15
4.1.1.9 ZapisywanieFaktyczne()	15
4.1.1.10 ZapisywanieTras()	15
4.1.1.11 Zapisz()	16
4.1.1.12 ZapiszdoPliku()	16
4.2 funkcje.h File Reference	17
4.2.1 Typedef Documentation	17
4.2.1.1 M	17
4.2.1.2 T	18
4.2.2 Function Documentation	18
4.2.2.1 AlgorytmDijkstry()	18
4.2.2.2 DodajdoDrzewa()	18
4.2.2.3 DodawanieDoListy()	18
4.2.2.4 odczytajParametry()	20
4.2.2.5 Usun()	20
4.2.2.6 UsunListy()	20
4.2.2.7 UsunStruktureTrasy()	21
4.2.2.8 WczytajDane()	21
4.2.2.9 ZapisywanieFaktyczne()	21
4.2.2.10 ZapisywanieTras()	22
4.2.2.11 Zapisz()	22
4.2.2.12 ZapiszdoPliku()	22
4.2.3 Variable Documentation	23
4.2.3.1 MAXINT	23
4.3 main.cpp File Reference	23
4.3.1 Function Documentation	23
4.3.1.1 main()	23
4.4 mem-leak-config.h File Reference	24
4.4.1 Macro Definition Documentation	24
4.4.1.1 TURN_ON_MEM_LEAK_DETECTION	24
4.5 mem-leak-detect.cpp File Reference	24
4.5.1 Function Documentation	24
4.5.1.1 calloc()	24
4.5.1.2 free()	25
4.5.1.3 malloc()	25
4.5.1.4 operator delete()	25
4.5.1.5 operator delete[]()	25

4.5.1.6 operator new() [1/2]	25
4.5.1.7 operator new() [2/2]	25
4.5.1.8 operator new[]() [1/2]	26
4.5.1.9 operator new[]() [2/2]	26
4.5.1.10 realloc()	26
4.6 mem-leak-detect.h File Reference	26
4.6.1 Macro Definition Documentation	27
4.6.1.1 calloc	27
4.6.1.2 DEBUG_NEW	27
4.6.1.3 free	27
4.6.1.4 malloc	27
4.6.1.5 new	27
4.6.1.6 realloc	27
4.6.2 Function Documentation	28
4.6.2.1 calloc()	28
4.6.2.2 free()	28
4.6.2.3 malloc()	28
4.6.2.4 operator delete()	28
4.6.2.5 operator delete[]()	28
4.6.2.6 operator new()	29
4.6.2.7 operator new[]()	29
4.6.2.8 realloc()	29
<b>Index</b>	<b>31</b>



# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Drzewo</a>	5
<a href="#">Odleglosc</a>	6
<a href="#">Parametry</a>	7
<a href="#">PorownajDrzewa</a>	8
<a href="#">Trasa</a>	9





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">funkcje.cpp</a>	11
<a href="#">funkcje.h</a>	17
<a href="#">main.cpp</a>	23
<a href="#">mem-leak-config.h</a>	24
<a href="#">mem-leak-detect.cpp</a>	24
<a href="#">mem-leak-detect.h</a>	26



## Chapter 3

# Class Documentation

### 3.1 Drzewo Struct Reference

```
#include <funkcje.h>
```

#### Public Attributes

- [M Miasto](#)  
*nazwa miasta przechowywanego w wezle*
- [Drzewo \\* pLewy](#)  
*wskaznik na lewy wezel drzewa*
- [Drzewo \\* pPrawy](#)  
*wskaznik na prawy wezel drzewa*
- [Odleglosc \\* pOdleglosci](#)  
*wskaznik na liste zawierajaca trasy wychodzace z danego miasta*
- `double current_dist`  
*obecna odleglosc od siedziby*
- [Drzewo \\* poprzednik\\_w\\_najkrotszej\\_trasie](#)  
*poprzednie odwiedzone miasto w danej trasie*

#### 3.1.1 Detailed Description

wezel drzewa poszukiwan binarnych

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 current\_dist

```
double Drzewo::current_dist
```

obecna odleglosc od siedziby

### 3.1.2.2 Miasto

`M Drzewo::Miasto`

nazwa miasta przechowywanego w wezle

### 3.1.2.3 pLewy

`Drzewo* Drzewo::pLewy`

wskaznik na lewy wezel drzewa

### 3.1.2.4 pOdleglosci

`Odleglosc* Drzewo::pOdleglosci`

wskaznik na liste zawierajaca trasy wychodzace z danego miasta

### 3.1.2.5 poprzednik\_w\_najkrotszej\_trasie

`Drzewo* Drzewo::poprzednik_w_najkrotszej_trasie`

poprzednie odwiedzone miasto w danej trasie

### 3.1.2.6 pPrawy

`Drzewo* Drzewo::pPrawy`

wskaznik na prawy wezel drzewa

The documentation for this struct was generated from the following file:

- [funkcje.h](#)

## 3.2 Odleglosc Struct Reference

```
#include <funkcje.h>
```

## Public Attributes

- [T](#) [kilometry](#)  
*odleglosc miedzy miastami*
- [Drzewo](#) \* [Polaczenie](#)  
*wskaznik odnoszacy sie do miasta z ktorym jest polaczenie*
- [Odleglosc](#) \* [pNastepny](#)  
*wskaznik na nastepny element listy*

### 3.2.1 Detailed Description

element listy jednokierunkowej

### 3.2.2 Member Data Documentation

#### 3.2.2.1 [kilometry](#)

[T](#) [Odleglosc::kilometry](#)

*odleglosc miedzy miastami*

#### 3.2.2.2 [pNastepny](#)

[Odleglosc](#)\* [Odleglosc::pNastepny](#)

*wskaznik na nastepny element listy*

#### 3.2.2.3 [Polaczenie](#)

[Drzewo](#)\* [Odleglosc::Polaczenie](#)

*wskaznik odnoszacy sie do miasta z ktorym jest polaczenie*

The documentation for this struct was generated from the following file:

- [funkcje.h](#)

## 3.3 Parametry Struct Reference

```
#include <funkcje.h>
```

## Public Attributes

- [M plikwejscowy](#)  
*nazwa pliku wejscowego*
- [M plikwyjscowy](#)  
*nazwa pliku wyjscowego*
- [M siedziba](#)  
*nazwa siedziby*

### 3.3.1 Detailed Description

struktura przechowująca odczytane parametry

### 3.3.2 Member Data Documentation

#### 3.3.2.1 plikwejscowy

[M Parametry::plikwejscowy](#)

nazwa pliku wejscowego

#### 3.3.2.2 plikwyjscowy

[M Parametry::plikwyjscowy](#)

nazwa pliku wyjscowego

#### 3.3.2.3 siedziba

[M Parametry::siedziba](#)

nazwa siedziby

The documentation for this struct was generated from the following file:

- [funkcje.h](#)

## 3.4 PorównajDrzewa Struct Reference

```
#include <funkcje.h>
```

## Public Member Functions

- `bool operator() (Drzewo *&d1, Drzewo *&d2)`

### 3.4.1 Detailed Description

struktura porownujaca elementy w kolejce priorytetowej

### 3.4.2 Member Function Documentation

#### 3.4.2.1 operator()

```
bool PorownajDrzewa::operator() (
    Drzewo *& d1,
    Drzewo *& d2 ) [inline]
```

The documentation for this struct was generated from the following file:

- [funkcje.h](#)

## 3.5 Trasa Struct Reference

```
#include <funkcje.h>
```

### Public Attributes

- [M Miasto](#)  
*nazwa miasta w trasie*
- [Trasa \\* Nastepny](#)  
*wskaznik na nastepny element listy (miasto w trasie)*

### 3.5.1 Detailed Description

struktura przechowujaca wyznaczone trasy

### 3.5.2 Member Data Documentation

### 3.5.2.1 Miasto

`M Trasa::Miasto`

nazwa miasta w trasie

### 3.5.2.2 Nastepny

`Trasa* Trasa::Nastepny`

wskaznik na nastepny element listy (miasto w trasie)

The documentation for this struct was generated from the following file:

- [funkcje.h](#)



## Chapter 4

# File Documentation

### 4.1 funkcje.cpp File Reference

```
#include "funkcje.h"
#include <iostream>
#include <queue>
```

#### Functions

- bool [odczytajParametry](#) (int argc, char \*\*argv, [Parametry](#) &p)
- [Drzewo](#) \* [DodajdoDrzewa](#) ([Drzewo](#) \*&pKorzen, const std::string &miasto)
- void [DodawanieDoListy](#) ([Drzewo](#) \*&w1, [Drzewo](#) \*w2, T km)
- void [UsunListy](#) ([Drzewo](#) \*&pKorzen)
- void [Usun](#) ([Drzewo](#) \*&pKorzen)
- void [AlgorytmDijkstry](#) ([Drzewo](#) \*&Korzen)
- bool [WczytajDane](#) (std::ifstream &is, [Drzewo](#) \*&Korzen)
- void [ZapisywanieTras](#) ([Drzewo](#) \*Korzen, [Trasa](#) \*T1, std::ostream &os)
- void [ZapisywanieFaktyczne](#) ([Drzewo](#) \*Korzen, [Trasa](#) \*T1, std::ostream &os)
- void [Zapisz](#) ([Drzewo](#) \*Korzen, [Trasa](#) \*T1, std::ostream &os)
- void [ZapiszdoPliku](#) (std::ostream &os, [Trasa](#) \*T1, [Drzewo](#) \*Korzen)
- void [UsunStruktureTrasy](#) ([Trasa](#) \*&T1)

#### 4.1.1 Function Documentation

##### 4.1.1.1 AlgorytmDijkstry()

```
void AlgorytmDijkstry (
    Drzewo *& Korzen )
```

Funkcja znajduje najkrotsze trasy z danej siedziby do innych miast za pomoca Algorytmu Dijkstry

## Parameters

<i>Korzen</i>	adres korzenia
---------------	----------------

**4.1.1.2 DodajdoDrzewa()**

```
Drzewo* DodajdoDrzewa (
    Drzewo *& pKorzen,
    const M & miasto )
```

Funkcja dodaje miasto do drzewa

## Parameters

<i>pKorzen</i>	adres korzenia
<i>miasto</i>	nazwa miasta w drzewie

## Returns

zwraca adres dodanego wezla

**4.1.1.3 DodawanieDoListy()**

```
void DodawanieDoListy (
    Drzewo *& w1,
    Drzewo * w2,
    T km )
```

Funkcja dodaje do listy trase laczaca jedno miasto z drugim oraz odleglosc miedzy nimi

## Parameters

<i>w1</i>	adres pierwszego miasta
<i>w2</i>	adres drugiego miasta
<i>km</i>	odleglosc miedzy miastami

**4.1.1.4 odczytajParametry()**

```
bool odczytajParametry (
    int argc,
```

```
char ** argv,  
Parametry & p )
```

Funkcja odczytuje parametry potrzebne do wykonania programu

**Parameters**

<i>argc</i>	Liczba parametrow
<i>argv</i>	Tablica parametrow
<i>p</i>	Struktura przechowujaca parametry na potrzeby programu

**Returns**

zwraca false gdy ilosc parametrow sie nie zgadza true gdy uda sie prawidlowo odczytac parametry

**4.1.1.5 Usun()**

```
void Usun (
    Drzewo *& pKorzen )
```

Funkcja usuwa miasta z drzewa

**Parameters**

<i>pKorzen</i>	adres korzenia
----------------	----------------

**4.1.1.6 UsunListy()**

```
void UsunListy (
    Drzewo * pKorzen )
```

Funkcja usuwa polaczenia miedzy miastami

**Parameters**

<i>pKorzen</i>	adres korzenia
----------------	----------------

**4.1.1.7 UsunStrukturyTrasy()**

```
void UsunStrukturyTrasy (
    Trasa *& T1 )
```

Funkcja usuwajaca strukture pomocnicza sluzaca do przechowywania tras

## Parameters

<i>T1</i>	wskaznik na glowe listy jednokierunkowej przechowujacej trasy
-----------	---

#### 4.1.1.8 WczytajDane()

```
bool WczytajDane (
    std::ifstream & is,
    Drzewo *& Korzen )
```

Funkcja wczytuje dane z pliku wejscowego

## Parameters

<i>is</i>	strumien z ktorego pobieramy dane
<i>Korzen</i>	wskaznik na korzen drzewa poszukiwan binarnych do ktorego bedziemy dodawac dane

## Returns

zwraca true gdy uda sie prawidlowo wczytac wszystkie dane, false w przypadku gdy dane w pliku sa nieprawidlowe

#### 4.1.1.9 ZapisywanieFaktyczne()

```
void ZapisywanieFaktyczne (
    Drzewo * Korzen,
    Trasa * T1,
    std::ostream & os )
```

Funkcja dzieki ktorej odwiedzimy kazdy wezel drzewa (oprocz siedziby) przy zapisywaniu tras

## Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

#### 4.1.1.10 ZapisywanieTras()

```
void ZapisywanieTras (
    Drzewo * Korzen,
```

```
Trasa * T1,  
std::ostream & os )
```

Funkcja dzięki której pominiemy siedzibę przy przechodzeniu po węzłach drzewa przy wyznaczaniu tras od danego węzła

#### Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w której bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

#### 4.1.1.11 Zapisz()

```
void Zapisz (  
    Drzewo * Korzen,  
    Trasa * T1,  
    std::ostream & os )
```

Funkcja zapisująca przebieg tras do pomocniczej struktury T1

#### Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w której bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

#### 4.1.1.12 ZapiszdoPliku()

```
void ZapiszdoPliku (  
    std::ostream & os,  
    Trasa * T1,  
    Drzewo * Korzen )
```

Funkcja zapisująca do pliku wyznaczony przebieg tras i ich dlugosc

#### Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w której bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

## 4.2 funkcje.h File Reference

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

### Classes

- struct [Drzewo](#)
- struct [Odleglosc](#)
- struct [Trasa](#)
- struct [Parametry](#)
- struct [PorownajDrzewa](#)

### Typedefs

- typedef int [T](#)
- typedef std::string [M](#)

### Functions

- bool [odczytajParametry](#) (int argc, char \*\*argv, [Parametry](#) &p)
- [Drzewo](#) \* [DodajdoDrzewa](#) ([Drzewo](#) \*&pKorzen, const [M](#) &miasto)
- void [DodawanieDoListy](#) ([Drzewo](#) \*&w1, [Drzewo](#) \*w2, [T](#) km)
- void [UsunListy](#) ([Drzewo](#) \*pKorzen)
- void [Usun](#) ([Drzewo](#) \*&pKorzen)
- void [AlgorytmDijkstry](#) ([Drzewo](#) \*&Korzen)
- bool [WczytajDane](#) (std::ifstream &is, [Drzewo](#) \*&Korzen)
- void [ZapisywanieTras](#) ([Drzewo](#) \*Korzen, [Trasa](#) \*T1, std::ostream &os)
- void [ZapisywanieFaktyczne](#) ([Drzewo](#) \*Korzen, [Trasa](#) \*T1, std::ostream &os)
- void [Zapisz](#) ([Drzewo](#) \*Korzen, [Trasa](#) \*T1, std::ostream &os)
- void [ZapiszdoPliku](#) (std::ostream &os, [Trasa](#) \*T1, [Drzewo](#) \*Korzen)
- void [UsunStruktureTrasy](#) ([Trasa](#) \*&T1)

### Variables

- const [T](#) [MAXINT](#) = 2147483647

#### 4.2.1 Typedef Documentation

##### 4.2.1.1 [M](#)

```
typedef std::string M
```

wartosc przechowujaca nazwe miast oraz parametrow

#### 4.2.1.2 T

```
typedef int T
```

wartosc przechowujaca odleglosci miedzy miastami

### 4.2.2 Function Documentation

#### 4.2.2.1 AlgorytmDijkstry()

```
void AlgorytmDijkstry (  
    Drzewo *& Korzen )
```

Funkcja znajduje najkrotsze trasy z danej siedziby do innych miast za pomoca Algorytmu Dijkstry

##### Parameters

<i>Korzen</i>	adres korzenia
---------------	----------------

#### 4.2.2.2 DodajdoDrzewa()

```
Drzewo* DodajdoDrzewa (  
    Drzewo *& pKorzen,  
    const M & miasto )
```

Funkcja dodaje miasto do drzewa

##### Parameters

<i>pKorzen</i>	adres korzenia
<i>miasto</i>	nazwa miasta w drzewie

##### Returns

zwraca adres dodanego wezla

#### 4.2.2.3 DodawanieDoListy()

```
void DodawanieDoListy (  
    Drzewo *& w1,
```



```
Drzewo * w2,  
T km )
```

Funkcja dodaje do listy trase laczaca jedno miasto z drugim oraz odleglosc miedzy nimi

**Parameters**

<i>w1</i>	adres pierwszego miasta
<i>w2</i>	adres drugiego miasta
<i>km</i>	odleglosc miedzy miastami

**4.2.2.4 odczytajParametry()**

```
bool odczytajParametry (
    int argc,
    char ** argv,
    Parametry & p )
```

Funkcja odczytuje parametry potrzebne do wykonania programu

**Parameters**

<i>argc</i>	Liczba parametrow
<i>argv</i>	Tablica parametrow
<i>p</i>	Struktura przechowujaca parametry na potrzeby programu

**Returns**

zwraca false gdy ilosc parametrow sie nie zgadza true gdy uda sie prawidlowo odczytac parametry

**4.2.2.5 Usun()**

```
void Usun (
    Drzewo *& pKorzen )
```

Funkcja usuwa miasta z drzewa

**Parameters**

<i>pKorzen</i>	adres korzenia
----------------	----------------

**4.2.2.6 UsunListy()**

```
void UsunListy (
    Drzewo * pKorzen )
```

Funkcja usuwa polaczenia miedzy miastami

## Parameters

<i>pKorzen</i>	adres korzenia
----------------	----------------

#### 4.2.2.7 UsunStruktureTrasy()

```
void UsunStruktureTrasy (
    Trasa *& T1 )
```

Funkcja usuwająca strukture pomocnicza służąca do przechowywania tras

## Parameters

<i>T1</i>	wskaznik na głowę listy jednokierunkowej przechowującej trasy
-----------	---

#### 4.2.2.8 WczytajDane()

```
bool WczytajDane (
    std::ifstream & is,
    Drzewo *& Korzen )
```

Funkcja wczytuje dane z pliku wejściowego

## Parameters

<i>is</i>	strumień z którego pobieramy dane
<i>Korzen</i>	wskaznik na korzeń drzewa poszukiwan binarnych do którego będziemy dodawać dane

## Returns

zwraca true gdy uda się prawidłowo wczytać wszystkie dane, false w przypadku gdy dane w pliku są nieprawidłowe

#### 4.2.2.9 ZapisywanieFaktyczne()

```
void ZapisywanieFaktyczne (
    Drzewo * Korzen,
    Trasa * T1,
    std::ostream & os )
```

Funkcja dzięki której odwiedzimy każdy węzeł drzewa (oprócz siedziby) przy zapisywaniu tras

## Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

**4.2.2.10 ZapisywanieTras()**

```
void ZapisywanieTras (
    Drzewo * Korzen,
    Trasa * T1,
    std::ostream & os )
```

Funkcja dzięki której pominiemy siedzibę przy przechodzeniu po węzłach drzewa przy wyznaczaniu tras od danego węzła

## Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

**4.2.2.11 Zapisz()**

```
void Zapisz (
    Drzewo * Korzen,
    Trasa * T1,
    std::ostream & os )
```

Funkcja zapisująca przebieg tras do pomocniczej struktury T1

## Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

**4.2.2.12 ZapiszdoPliku()**

```
void ZapiszdoPliku (
    std::ostream & os,
```

```
Trasa * Tl,  
Drzewo * Korzen )
```

Funkcja zapisująca do pliku wyznaczony przebieg tras i ich dlugosc

#### Parameters

<i>Korzen</i>	wskaznik na korzen drzewa
<i>T1</i>	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
<i>os</i>	strumien sluzacy do zapisu tras do pliku

## 4.2.3 Variable Documentation

### 4.2.3.1 MAXINT

```
const T MAXINT = 2147483647
```

najwieksza mozliwa wartosc int

## 4.3 main.cpp File Reference

```
#include <iostream>  
#include "mem-leak-detect.h"  
#include "funkcje.h"  
#include <fstream>  
#include <queue>
```

### Functions

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

### 4.3.1 Function Documentation

#### 4.3.1.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

## 4.4 mem-leak-config.h File Reference

### Macros

- `#define` [TURN\\_ON\\_MEM\\_LEAK\\_DETECTION](#)

### 4.4.1 Macro Definition Documentation

#### 4.4.1.1 TURN\_ON\_MEM\_LEAK\_DETECTION

```
#define TURN_ON_MEM_LEAK_DETECTION
```

## 4.5 mem-leak-detect.cpp File Reference

```
#include "mem-leak-config.h"  
#include <unordered_map>  
#include <string>
```

### Functions

- `void *` [operator new](#) (size\_t size, const char \*file, uint32\_t line)
- `void *` [malloc](#) (size\_t size, const char \*file, uint32\_t line)
- `void *` [calloc](#) (size\_t num, size\_t size, const char \*file, uint32\_t line)
- `void *` [realloc](#) (void \*ptr, size\_t size, const char \*file, uint32\_t line)
- `void` [free](#) (void \*ptr, int)
- `void *` [operator new\[\]](#) (size\_t size)
- `void *` [operator new\[\]](#) (size\_t size, const char \*file, uint32\_t line)
- `void` [operator delete](#) (void \*ptr)
- `void *` [operator new](#) (size\_t size)
- `void` [operator delete\[\]](#) (void \*ptr)

### 4.5.1 Function Documentation

#### 4.5.1.1 calloc()

```
void* calloc (  
    size_t num,  
    size_t size,  
    const char * file,  
    uint32_t line )
```

#### 4.5.1.2 free()

```
void free (
    void * ptr,
    int )
```

#### 4.5.1.3 malloc()

```
void* malloc (
    size_t size,
    const char * file,
    uint32_t line )
```

#### 4.5.1.4 operator delete()

```
void operator delete (
    void * ptr )
```

#### 4.5.1.5 operator delete[]()

```
void operator delete[] (
    void * ptr )
```

#### 4.5.1.6 operator new() [1/2]

```
void* operator new (
    size_t size )
```

#### 4.5.1.7 operator new() [2/2]

```
void* operator new (
    size_t size,
    const char * file,
    uint32_t line )
```

#### 4.5.1.8 operator new[]() [1/2]

```
void* operator new[] (
    size_t size )
```

#### 4.5.1.9 operator new[]() [2/2]

```
void* operator new[] (
    size_t size,
    const char * file,
    uint32_t line )
```

#### 4.5.1.10 realloc()

```
void* realloc (
    void * ptr,
    size_t size,
    const char * file,
    uint32_t line )
```

## 4.6 mem-leak-detect.h File Reference

```
#include "mem-leak-config.h"
#include <cstdlib>
#include <stdint>
#include <stddef>
```

### Macros

- #define `DEBUG_NEW new`(\_\_FILE\_\_, \_\_LINE\_\_)
- #define `new DEBUG_NEW`
- #define `malloc(x)` malloc((x), \_\_FILE\_\_, \_\_LINE\_\_)
- #define `free(x)` free((x), 0)
- #define `calloc(x, y)` calloc((x), (y), \_\_FILE\_\_, \_\_LINE\_\_)
- #define `realloc(x, y)` realloc((x), (y), \_\_FILE\_\_, \_\_LINE\_\_)

### Functions

- void \* `operator new` (size\_t size, const char \*file, uint32\_t line)
- void \* `operator new[]` (size\_t size, const char \*file, uint32\_t line)
- void `operator delete` (void \*ptr)
- void `operator delete[]` (void \*ptr)
- void \* `malloc` (size\_t size, const char \*file, uint32\_t line)
- void \* `calloc` (size\_t num, size\_t size, const char \*file, uint32\_t line)
- void \* `realloc` (void \*ptr, size\_t size, const char \*file, uint32\_t line)
- void `free` (void \*ptr, int)



## 4.6.1 Macro Definition Documentation

### 4.6.1.1 calloc

```
#define calloc(  
    x,  
    y ) calloc((x), (y), __FILE__, __LINE__)
```

### 4.6.1.2 DEBUG\_NEW

```
#define DEBUG_NEW new(__FILE__, __LINE__)
```

### 4.6.1.3 free

```
#define free(  
    x ) free((x), 0)
```

### 4.6.1.4 malloc

```
#define malloc(  
    x ) malloc((x), __FILE__, __LINE__)
```

### 4.6.1.5 new

```
#define new DEBUG_NEW
```

### 4.6.1.6 realloc

```
#define realloc(  
    x,  
    y ) realloc((x), (y), __FILE__, __LINE__)
```

## 4.6.2 Function Documentation

### 4.6.2.1 calloc()

```
void* calloc (
    size_t num,
    size_t size,
    const char * file,
    uint32_t line )
```

### 4.6.2.2 free()

```
void free (
    void * ptr,
    int )
```

### 4.6.2.3 malloc()

```
void* malloc (
    size_t size,
    const char * file,
    uint32_t line )
```

### 4.6.2.4 operator delete()

```
void operator delete (
    void * ptr )
```

### 4.6.2.5 operator delete[]()

```
void operator delete[] (
    void * ptr )
```

#### 4.6.2.6 operator new()

```
void* operator new (
    size_t size,
    const char * file,
    uint32_t line )
```

#### 4.6.2.7 operator new[]()

```
void* operator new[] (
    size_t size,
    const char * file,
    uint32_t line )
```

#### 4.6.2.8 realloc()

```
void* realloc (
    void * ptr,
    size_t size,
    const char * file,
    uint32_t line )
```



# Index

- AlgorytmDijkstry
  - funkcje.cpp, [11](#)
  - funkcje.h, [18](#)
- calloc
  - mem-leak-detect.cpp, [24](#)
  - mem-leak-detect.h, [27](#), [28](#)
- current\_dist
  - Drzewo, [5](#)
- DEBUG\_NEW
  - mem-leak-detect.h, [27](#)
- DodajdoDrzewa
  - funkcje.cpp, [12](#)
  - funkcje.h, [18](#)
- DodawanieDoListy
  - funkcje.cpp, [12](#)
  - funkcje.h, [18](#)
- Drzewo, [5](#)
  - current\_dist, [5](#)
  - Miasto, [5](#)
  - pLewy, [6](#)
  - pOdleglosci, [6](#)
  - poprzednik\_w\_najkrotszej\_trasie, [6](#)
  - pPrawy, [6](#)
- free
  - mem-leak-detect.cpp, [24](#)
  - mem-leak-detect.h, [27](#), [28](#)
- funkcje.cpp, [11](#)
  - AlgorytmDijkstry, [11](#)
  - DodajdoDrzewa, [12](#)
  - DodawanieDoListy, [12](#)
  - odczytajParametry, [12](#)
  - Usun, [14](#)
  - UsunListy, [14](#)
  - UsunStruktureTrasy, [14](#)
  - WczytajDane, [15](#)
  - ZapisywanieFaktyczne, [15](#)
  - ZapisywanieTras, [15](#)
  - Zapisz, [16](#)
  - ZapiszdoPliku, [16](#)
- funkcje.h, [17](#)
  - AlgorytmDijkstry, [18](#)
  - DodajdoDrzewa, [18](#)
  - DodawanieDoListy, [18](#)
  - M, [17](#)
  - MAXINT, [23](#)
  - odczytajParametry, [20](#)
  - T, [17](#)
  - Usun, [20](#)
  - UsunListy, [20](#)
  - UsunStruktureTrasy, [21](#)
  - WczytajDane, [21](#)
  - ZapisywanieFaktyczne, [21](#)
  - ZapisywanieTras, [22](#)
  - Zapisz, [22](#)
  - ZapiszdoPliku, [22](#)
- kilometry
  - Odleglosc, [7](#)
- M
  - funkcje.h, [17](#)
- main
  - main.cpp, [23](#)
- main.cpp, [23](#)
  - main, [23](#)
- malloc
  - mem-leak-detect.cpp, [25](#)
  - mem-leak-detect.h, [27](#), [28](#)
- MAXINT
  - funkcje.h, [23](#)
- mem-leak-config.h, [24](#)
  - TURN\_ON\_MEM\_LEAK\_DETECTION, [24](#)
- mem-leak-detect.cpp, [24](#)
  - calloc, [24](#)
  - free, [24](#)
  - malloc, [25](#)
  - operator delete, [25](#)
  - operator delete[], [25](#)
  - operator new, [25](#)
  - operator new[], [25](#), [26](#)
  - realloc, [26](#)
- mem-leak-detect.h, [26](#)
  - calloc, [27](#), [28](#)
  - DEBUG\_NEW, [27](#)
  - free, [27](#), [28](#)
  - malloc, [27](#), [28](#)
  - new, [27](#)
  - operator delete, [28](#)
  - operator delete[], [28](#)
  - operator new, [28](#)
  - operator new[], [29](#)
  - realloc, [27](#), [29](#)
- Miasto
  - Drzewo, [5](#)
  - Trasa, [9](#)
- Nastepny

- Trasa, 10
- new
  - mem-leak-detect.h, 27
- odczytajParametry
  - funkcje.cpp, 12
  - funkcje.h, 20
- Odleglosc, 6
  - kilometry, 7
  - pNastepny, 7
  - Polaczenie, 7
- operator delete
  - mem-leak-detect.cpp, 25
  - mem-leak-detect.h, 28
- operator delete[]
  - mem-leak-detect.cpp, 25
  - mem-leak-detect.h, 28
- operator new
  - mem-leak-detect.cpp, 25
  - mem-leak-detect.h, 28
- operator new[]
  - mem-leak-detect.cpp, 25, 26
  - mem-leak-detect.h, 29
- operator()
  - PorownajDrzewa, 9
- Parametry, 7
  - plikwejscowy, 8
  - plikwyjscowy, 8
  - siedziba, 8
- pLewy
  - Drzewo, 6
- plikwejscowy
  - Parametry, 8
- plikwyjscowy
  - Parametry, 8
- pNastepny
  - Odleglosc, 7
- pOdleglosci
  - Drzewo, 6
- Polaczenie
  - Odleglosc, 7
- poprzednik\_w\_najkrotszej\_trasie
  - Drzewo, 6
- PorownajDrzewa, 8
  - operator(), 9
- pPrawy
  - Drzewo, 6
- realloc
  - mem-leak-detect.cpp, 26
  - mem-leak-detect.h, 27, 29
- siedziba
  - Parametry, 8
- T
  - funkcje.h, 17
- Trasa, 9
  - Miasto, 9
  - Nastepny, 10
- TURN\_ON\_MEM\_LEAK\_DETECTION
  - mem-leak-config.h, 24
- Usun
  - funkcje.cpp, 14
  - funkcje.h, 20
- UsunListy
  - funkcje.cpp, 14
  - funkcje.h, 20
- UsunStruktureTrasy
  - funkcje.cpp, 14
  - funkcje.h, 21
- WczytajDane
  - funkcje.cpp, 15
  - funkcje.h, 21
- ZapisywanieFaktyczne
  - funkcje.cpp, 15
  - funkcje.h, 21
- ZapisywanieTras
  - funkcje.cpp, 15
  - funkcje.h, 22
- Zapisz
  - funkcje.cpp, 16
  - funkcje.h, 22
- ZapiszdoPliku
  - funkcje.cpp, 16
  - funkcje.h, 22