My Project

Generated by Doxygen 1.8.16

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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 plikwejsciowy
  nazwa pliku wejsciowego
- M plikwyjsciowy
  nazwa pliku wyjsciowego
- M siedziba
  nazwa siedziby

3.3.1 Detailed Description

struktura przechowujaca odczytane parametry

3.3.2 Member Data Documentation

3.3.2.1 plikwejsciowy

`M Parametry::plikwejsciowy`

nazwa pliku wejsciowego

3.3.2.2 plikwyjsciowy

`M Parametry::plikwyjsciowy`

nazwa pliku wyjsciowego

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 PorownajDrzewa 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

Korzen	adres korzenia

4.1.1.2 DodajdoDrzewa()

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

Funkcja dodaje miasto do drzewa

Parameters

pKorzen	adres korzenia
miasto	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

w1	adres pierwszego miasta
w2	adres drugiego miasta
km	odleglosc miedzy miastami

4.1.1.4 odczytajParametry()

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

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

Funkcja odczytuje parametry potrzebne do wykonania programu

Parameters

argc	Liczba parametrow
argv	Tablica parametrow
p	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

pKorzen	adres korzenia

4.1.1.6 UsunListy()

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

Funkcja usuwa polaczenia miedzy miastami

Parameters

pKorzen	adres korzenia

4.1.1.7 UsunStruktureTrasy()

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

Funkcja usuwajaca strukture pomocnicza sluzaca do przechowywania tras

Parameters

T1	wskaznik na glowe listy jednokierunkowej przechowujacej trasy

4.1.1.8 WczytajDane()

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

Funkcja wczytuje dane z pliku wejsciowego

Parameters

is	strumien z ktorego pobieramy dane
Korzen	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

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

4.1.1.10 ZapisywanieTras()

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

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

Funkcja dzieki ktorej pominiemy siedzibe przy przechodzeniu po wezlach drzewa przy wyznaczaniu tras od danego wezla

Parameters

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

4.1.1.11 Zapisz()

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

Funkcja zapisujaca przebieg tras do pomocniczej struktury T1

Parameters

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

4.1.1.12 ZapiszdoPliku()

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

Funkcja zapisujaca do pliku wyznaczony przebieg tras i ich dlugosc

Parameters

Korzen	wskaznik na korzen drzewa
T1	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
os	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

Korzen	adres korzenia

4.2.2.2 DodajdoDrzewa()

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

Funkcja dodaje miasto do drzewa

Parameters

pKorzen	adres korzenia
miasto	nazwa miasta w drzewie

Returns

zwraca adres dodanego wezla

4.2.2.3 DodawanieDoListy()

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

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

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

Parameters

w1	adres pierwszego miasta
w2	adres drugiego miasta
km	odleglosc miedzy miastami

4.2.2.4 odczytajParametry()

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

Funkcja odczytuje parametry potrzebne do wykonania programu

Parameters

argc	Liczba parametrow
argv	Tablica parametrow
p	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

pKorzen	adres korzenia

4.2.2.6 UsunListy()

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

Funkcja usuwa polaczenia miedzy miastami

Parameters

pKorzen	adres korzenia

4.2.2.7 UsunStruktureTrasy()

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

Funkcja usuwajaca strukture pomocnicza sluzaca do przechowywania tras

Parameters

T1	wskaznik na glowe listy jednokierunkowej przechowujacej trasy

4.2.2.8 WczytajDane()

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

Funkcja wczytuje dane z pliku wejsciowego

Parameters

is	strumien z ktorego pobieramy dane
Korzen	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

4.2.2.9 ZapisywanieFaktyczne()

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

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

Parameters

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

4.2.2.10 ZapisywanieTras()

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

Funkcja dzieki ktorej pominiemy siedzibe przy przechodzeniu po wezlach drzewa przy wyznaczaniu tras od danego wezla

Parameters

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

4.2.2.11 Zapisz()

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

Funkcja zapisujaca przebieg tras do pomocniczej struktury T1

Parameters

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

4.2.2.12 ZapiszdoPliku()

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

```
            Trasa * T1,
            Drzewo * Korzen )
```

Funkcja zapisujaca do pliku wyznaczony przebieg tras i ich dlugosc

Parameters

Korzen	wskaznik na korzen drzewa
T1	struktura pomocnicza w ktorej bedziemy przechowywac wyznaczone trasy
os	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 <cstdint>
#include <cstddef>
```

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