Jarvis-Patrick Clustering

Wygenerowano przez Doxygen 1.9.6

Rozdział 1

Indeks klas

1.1 Lista klas

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

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 3

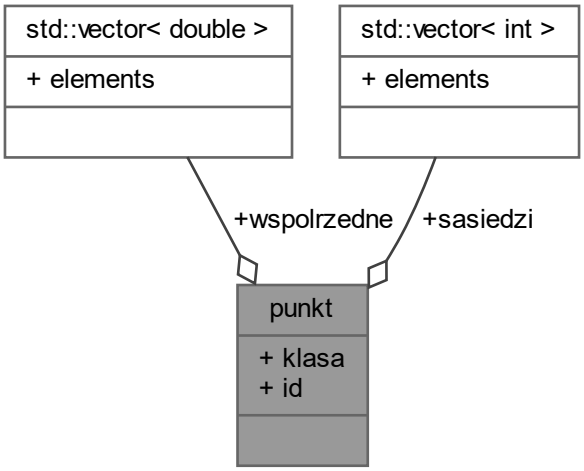
Dokumentacja klas

3.1 Dokumentacja struktury punkt

Typ zlozony przechowujacy informacje o kazdym wczytanym punkcie.

`#include <struktury.h>`

Diagram współpracy dla punkt:



Atrybuty publiczne

- std::vector< double > wspolrzedne
- int klasa
- int id
- std::vector< int > sasiedzi

3.1.1 Opis szczegółowy

Typ zlozony przechowujacy informacje o kazdym wczytanym punkcie.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 id

```
int punkt::id
```

Przechowuje id punktu

3.1.2.2 klasa

```
int punkt::klasa
```

Przechowuje klase punktu

3.1.2.3 sasiedzi

```
std::vector<int> punkt::sasiedzi
```

Przechowuje id n najblizszych sasiadow, gdzie n wczytywane z parametrow wejsciowych

3.1.2.4 wspolrzedne

```
std::vector<double> punkt::wspolrzedne
```

Przechowuje wspolrzedne punktu

Dokumentacja dla tej struktury została wygenerowana z pliku:

- struktury.h

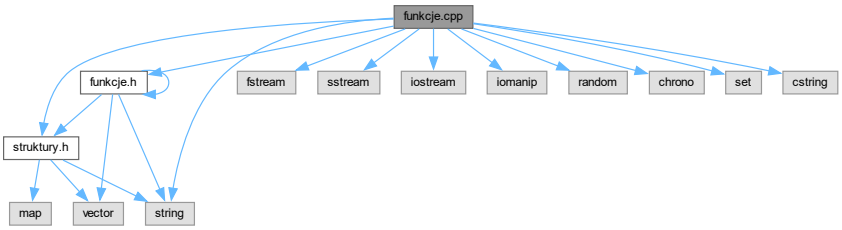
Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku funkcje.cpp

```
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <random>
#include <chrono>
#include <set>
#include <cstring>
#include "struktury.h"
#include "funkcje.h"
```

Wykres zależności załączania dla funkcje.cpp:



Funkcje

- std::map< std::string, std::string > parametry (const int &ile, char *params[])

  Funkcja sprawdza poprawnosc parametrow wejsciowych.
- void poradnik ()

  Funkcja wypisujaca na ekran poradnik dotyczacy uzytkowania programu.
- void poradnik_wejscie ()

  Funkcja informujaca o niepoprawnych typach danych wejsciowych.
- std::vector< punkt > wczytywanie (const std::string &input)

  Funkcja wczytujaca punkty z pliku wejsciowego.

- void przypisz_sasiadow (std::vector< punkt > &punkty, const int &n)

  Funkcja ustalajaca najblizszych sasiadow kazdego z punktow(modyfikuje istniejacy wektor)
- double obl_odl (const punkt &i, const punkt &j)

  Funkcja obliczajaca odleglosc miedzy dwoma punktami.
- void przypisz_klase (std::vector< punkt > &punkty, const int &n, const int &w)

  Funkcja przypisujaca klase kazdemu z punktow zgodnie z glownym zalozeniem algorytmu.
- void klasa_temp (std::vector< punkt > &punkty)

  Funkcja przypisujaca tymczasowa klase kazdemu punktowi (modyfikuje istniejacy wektor)
- void zapisz_do_pliku (const std::vector< punkt > &punkty, const std::string &output)

  Funkcja wypisujaca klasy punktow oraz ich wspolrzedne do pliku wyjsciowego.

4.1.1 Dokumentacja funkcji

4.1.1.1 klasa_temp()

```
void klasa_temp (
            std::vector< punkt > & punkty )
```

Funkcja przypisujaca tymczasowa klase kazdemu punktowi (modyfikuje istniejacy wektor)

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow

Oto graf wywoływań tej funkcji:

4.1.1.2 obl_odl()

```
double obl_odl (
            const punkt & i,
            const punkt & j )
```

Funkcja obliczajaca odleglosc miedzy dwoma punktami.

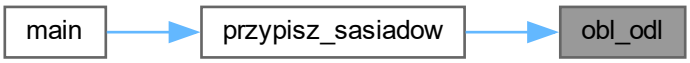
Parametry

i	Pierwszy punkt
n	Drugi punkt

Zwraca

Zwraca odleglosc euklidesowa

Oto graf wywoływań tej funkcji:



4.1.1.3 parametry()

```
std::map< std::string, std::string > parametry (
            const int & ile,
            char * params[] )
```

Funkcja sprawdza poprawnosc parametrow wejsciowych.

Parametry

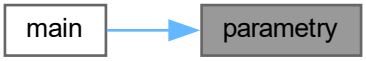
ile	Liczba parametrow wejsciowych programu
params[]	Tablica zawierajaca te parametry

Zwraca

Zwraca mape parametrow

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

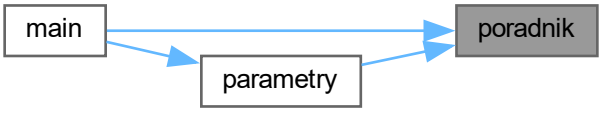


4.1.1.4 poradnik()

```
void poradnik ( )
```

Funkcja wypisujaca na ekran poradnik dotyczacy uzytkowania programu.

Oto graf wywoływań tej funkcji:



4.1.1.5 poradnik_wejscie()

```
void poradnik_wejscie ( )
```

Funkcja informujaca o niepoprawnych typach danych wejsciowych.

Zwraca

Nie zwraca niczego

Oto graf wywoływań tej funkcji:

4.1.1.6 przypisz_klase()

```
void przypisz_klase (
            std::vector< punkt > & punkty,
            const int & n,
            const int & w )
```

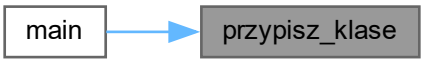
Funkcja przypisujaca klase kazdemu z punktow zgodnie z glownym zalozeniem algorytmu.

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow
	n	Liczba najblizszych sasiadow przekazana jako parametr n (int liczba; double ilosc;)
	w	Liczba wspolnych sasiadow miedzy dowlonymi punktami przekazana jako parametr w

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:



4.1.1.7 przypisz_sasiadow()

```
void przypisz_sasiadow (
            std::vector< punkt > & punkty,
            const int & n )
```

Funkcja ustalajaca najblizszych sasiadow kazdego z punktow(modyfikuje istniejacy wektor)

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow
	n	Liczba najblizszych sasiadow przekazana jako parametr n

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

4.1.1.8 wczytywanie()

```
std::vector< punkt > wczytywanie (
            const std::string & input )
```

Funkcja wczytujaca punkty z pliku wejsciowego.

Parametry

input	Nazwa pliku wejsciowego przekazanego jako parametr i

Zwraca

Zwraca wektor wczytanych punktow

Oto graf wywoływań tej funkcji:

4.1.1.9 zapisz_do_pliku()

```
void zapisz_do_pliku (
            const std::vector< punkt > & punkty,
            const std::string & output )
```

Funkcja wypisujaca klasy punktow oraz ich wspolrzedne do pliku wyjsciowego.

Parametry

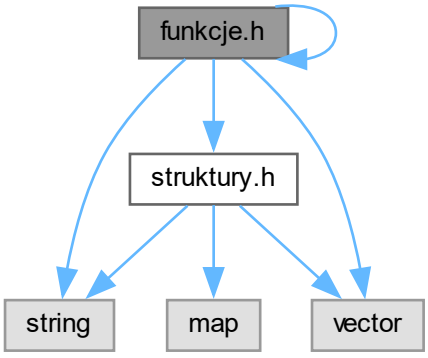
out	*punkty*	Wektor wszystkich pobranych punktow
	output	Nazwa pliku wyjsciowego przekazana jako parametr o

Oto graf wywoływań tej funkcji:

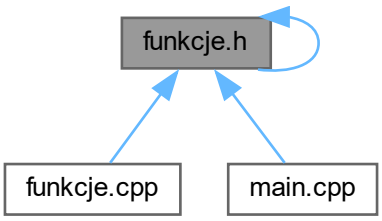
4.2 Dokumentacja pliku funkcje.h

```
#include <string>
#include <vector>
#include "struktury.h"
#include "funkcje.h"
```

Wykres zależności załączania dla funkcje.h:



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



Funkcje

- std::map< std::string, std::string > parametry (const int &ile, char *params[])

  Funkcja sprawdza poprawnosc parametrow wejsciowych.
- void poradnik ()

  Funkcja wypisujaca na ekran poradnik dotyczacy uzytkowania programu.
- void poradnik_wejscie ()

  Funkcja informujaca o niepoprawnych typach danych wejsciowych.
- std::vector< punkt > wczytywanie (const std::string &input)

  Funkcja wczytujaca punkty z pliku wejsciowego.
- void przypisz_sasiadow (std::vector< punkt > &punkty, const int &n)

  Funkcja ustalajaca najblizszych sasiadow kazdego z punktow(modyfikuje istniejacy wektor)
- double obl_odl (const punkt &i, const punkt &j)

  Funkcja obliczajaca odleglosc miedzy dwoma punktami.
- void przypisz_klase (std::vector< punkt > &punkty, const int &n, const int &w)

  Funkcja przypisujaca klase kazdemu z punktow zgodnie z glownym zalozeniem algorytmu.
- void klasa_temp (std::vector< punkt > &punkty)

  Funkcja przypisujaca tymczasowa klase kazdemu punktowi (modyfikuje istniejacy wektor)
- void zapisz_do_pliku (const std::vector< punkt > &punkty, const std::string &output)

  Funkcja wypisujaca klasy punktow oraz ich wspolrzedne do pliku wyjsciowego.

4.2.1 Dokumentacja funkcji

4.2.1.1 klasa_temp()

```
void klasa_temp (
            std::vector< punkt > & punkty )
```

Funkcja przypisujaca tymczasowa klase kazdemu punktowi (modyfikuje istniejacy wektor)

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow

Oto graf wywoływań tej funkcji:

4.2.1.2 obl_odl()

```
double obl_odl (
            const punkt & i,
            const punkt & j )
```

Funkcja obliczajaca odleglosc miedzy dwoma punktami.

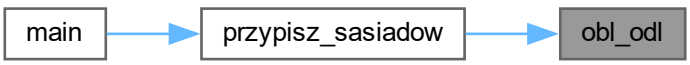
Parametry

i	Pierwszy punkt
n	Drugi punkt

Zwraca

Zwraca odleglosc euklidesowa

Oto graf wywoływań tej funkcji:



4.2.1.3 parametry()

```
std::map< std::string, std::string > parametry (
            const int & ile,
            char * params[] )
```

Funkcja sprawdza poprawnosc parametrow wejsciowych.

Parametry

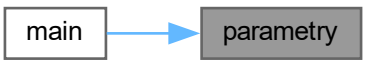
ile	Liczba parametrow wejsciowych programu
params[]	Tablica zawierajaca te parametry

Zwraca

Zwraca mape parametrow

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

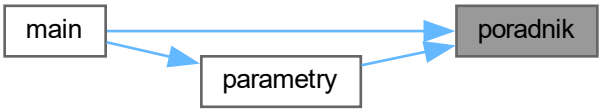


4.2.1.4 poradnik()

```
void poradnik ( )
```

Funkcja wypisujaca na ekran poradnik dotyczacy uzytkowania programu.

Oto graf wywoływań tej funkcji:



4.2.1.5 poradnik_wejscie()

```
void poradnik_wejscie ( )
```

Funkcja informujaca o niepoprawnych typach danych wejsciowych.

Zwraca

Nie zwraca niczego

Oto graf wywoływań tej funkcji:

4.2.1.6 przypisz_klase()

```
void przypisz_klase (
            std::vector< punkt > & punkty,
            const int & n,
            const int & w )
```

Funkcja przypisujaca klase kazdemu z punktow zgodnie z glownym zalozeniem algorytmu.

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow
	n	Liczba najblizszych sasiadow przekazana jako parametr n (int liczba; double ilosc;)
	w	Liczba wspolnych sasiadow miedzy dowlonymi punktami przekazana jako parametr w

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

4.2.1.7 przypisz_sasiadow()

```
void przypisz_sasiadow (
            std::vector< punkt > & punkty,
            const int & n )
```

Funkcja ustalajaca najblizszych sasiadow kazdego z punktow(modyfikuje istniejacy wektor)

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow
	n	Liczba najblizszych sasiadow przekazana jako parametr n

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

4.2.1.8 wczytywanie()

```
std::vector< punkt > wczytywanie (
            const std::string & input )
```

Funkcja wczytujaca punkty z pliku wejsciowego.

Parametry

input	Nazwa pliku wejsciowego przekazanego jako parametr i

Zwraca

Zwraca wektor wczytanych punktow

Oto graf wywoływań tej funkcji:

4.2.1.9 zapisz_do_pliku()

```
void zapisz_do_pliku (
            const std::vector< punkt > & punkty,
            const std::string & output )
```

Funkcja wypisujaca klasy punktow oraz ich wspolrzedne do pliku wyjsciowego.

Parametry

out	*punkty*	Wektor wszystkich pobranych punktow
	output	Nazwa pliku wyjsciowego przekazana jako parametr o

Oto graf wywoływań tej funkcji:

4.3 funkcje.h

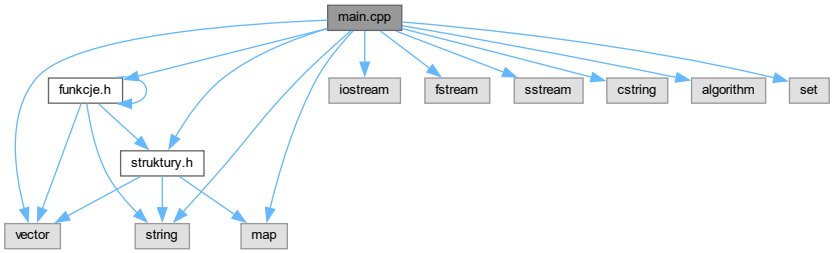
Idź do dokumentacji tego pliku.

```
00001
00003 // DECLARATIONS (HEADERS) OF FUNCTIONS
00004
00005 #ifndef FUNKCJE_H
00006 #define FUNKCJE_H
00007
00008 #include <string>
00009 #include <vector>
00010
00011 #include "struktury.h"
00012 #include "funkcje.h"
00013
00019 std::map<std::string, std::string> parametry(const int & ile, char * params[]);
00020
00023 void poradnik();
00024
00028 void poradnik_wejscie();
00029
00034 std::vector<punkt> wczytywanie(const std::string& input);
00035
00040 void przypisz_sasiadow(std::vector<punkt>& punkty, const int& n);
00041
00048 double obl_odl(const punkt& i, const punkt& j);
00049
00055 void przypisz_klase(std::vector<punkt>& punkty, const int& n, const int& w);
00056
00061 void klasa_temp(std::vector<punkt>& punkty);
00062
00067 void zapisz_do_pliku(const std::vector<punkt>& punkty, const std::string& output);
00068
00069 #endif
```

4.4 Dokumentacja pliku main.cpp

```
#include <vector>
#include <string>
#include <iostream>
#include <map>
#include <fstream>
#include <sstream>
#include <cstring>
#include <algorithm>
#include <set>
#include "struktury.h"
#include "funkcje.h"
```

Wykres zależności załączania dla main.cpp:



Funkcje

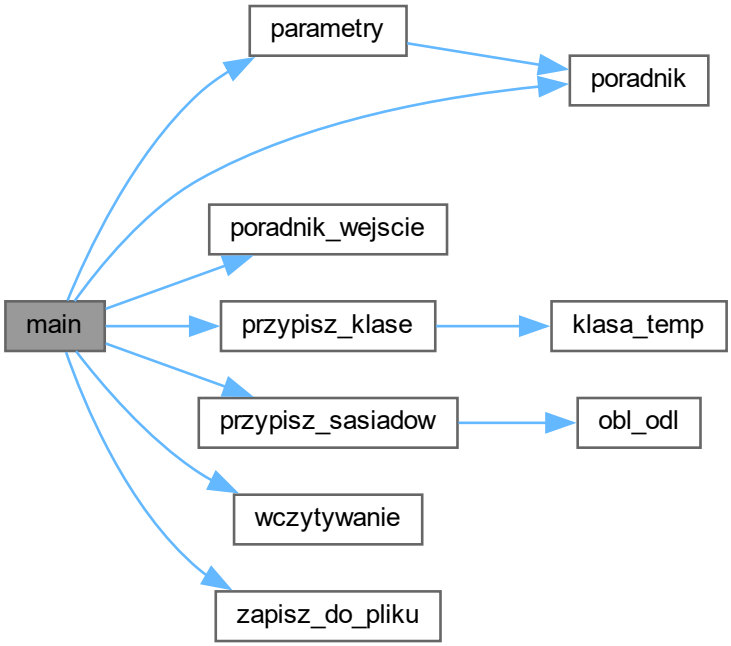
- int main (int ile, char ∗params[ ])

4.4.1 Dokumentacja funkcji

4.4.1.1 main()

```
int main (
            int ile,
            char * params[] )
```

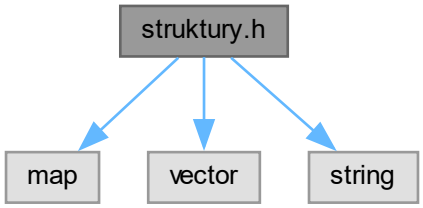
Oto graf wywołań dla tej funkcji:



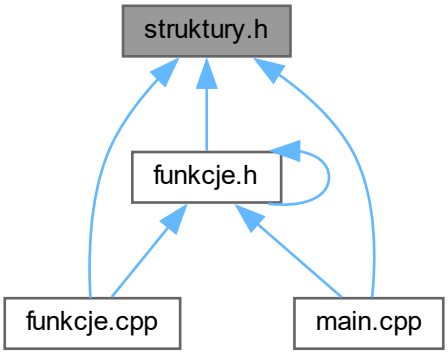
4.5 Dokumentacja pliku struktury.h

```
#include <map>
#include <vector>
#include <string>
```

Wykres zależności załączania dla struktury.h:



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



Komponenty

- struct punkt

  Typ zlozony przechowujacy informacje o kazdym wczytanym punkcie.

4.6 struktury.h

Idź do dokumentacji tego pliku.

```
00001
00003 #ifndef STRUKTURY_H
00004 #define STRUKTURY_H
00005
00006 #include <map>
00007 #include <vector>
00008 #include <string>
00012 struct punkt
00013 {
00015     std::vector<double> wspolrzedne;
00017     int klasa;
00019     int id;
00021     std::vector<int> sasiedzi;
00022
00023     // std::set
00024     // struktura danych rozłącznych
00025
00026 };
00027
00028 #endif
```