B-Drzewo

Wygenerowano przez Doxygen 1.9.1

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 komenda

Struktura komendy.

`#include <struktury.h>`

Atrybuty publiczne

- **typ rodzaj**
- std::vector< double > **params**
  rodzaj komendy
- std::string **nazwa_pliku**
  parametry liczbowe

3.1.1 Opis szczegółowy

Struktura komendy.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 nazwa_pliku

`std::string komenda::nazwa_pliku`

parametry liczbowe

3.1.2.2 params

```
std::vector<double> komenda::params
```

rodzaj komendy

3.1.2.3 rodzaj

```
typ komenda::rodzaj
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- **struktury.h**

3.2 Dokumentacja struktury wezel

Struktura wezla drzewa.

```
#include <drzewo.h>
```

Atrybuty publiczne

- std::list< double > **wartosci**
- std::list< std::shared_ptr< **wezel** > > **potomki**
  Wartosci w wezle.
- std::weak_ptr< **wezel** > **pRodzic**
  Potomki wezla.
- std::shared_ptr< **wezel** > **wskaznik_na_nowy_korzen** = nullptr
  Rodzic wezla.
- bool **czyLisc** {true}
  Wskaznik ktory przechowuje adres na nowy korzen, gdy dokonujemy jego zmiany.
- bool **czyKorzenDoZmiany** {false}
  Zmienna okreslajaca czy wezel jest lisciem.

Statyczne atrybuty publiczne

- static double **rzad**
  Zmienna okreslajaca czy dany wezel jest korzeniem ktory musimy zmienic na nowy.

3.2.1 Opis szczegółowy

Struktura wezla drzewa.

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 czyKorzenDoZmiany

```
bool wezel::czyKorzenDoZmiany {false}
```

Zmienna okreslajaca czy wezel jest lisciem.

3.2.2.2 czyLisc

```
bool wezel::czyLisc {true}
```

Wskaznik ktory przechowuje adres na nowy korzen, gdy dokonujemy jego zmiany.

3.2.2.3 potomki

```
std::list<std::shared_ptr< wezel> > wezel::potomki
```

Wartosci w wezle.

3.2.2.4 pRodzic

```
std::weak_ptr< wezel> wezel::pRodzic
```

Potomki wezla.

3.2.2.5 rzad

```
double wezel::rzad  [static]
```

Zmienna okreslajaca czy dany wezel jest korzeniem ktory musimy zmienic na nowy.

3.2.2.6 wartosci

```
std::list<double> wezel::wartosci
```

3.2.2.7 wskaznik_na_nowy_korzen

```
std::shared_ptr< wezel> wezel::wskaznik_na_nowy_korzen = nullptr
```

Rodzic wezla.

Dokumentacja dla tej struktury została wygenerowana z plików:

- **drzewo.h**
- **drzewo.cpp**

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku drzewo.cpp

```
#include <iostream>
#include <fstream>
#include "funkcje.h"
#include "struktury.h"
#include "drzewo.h"
#include <tuple>
```

Definicje

- #define **debug**(x) std::cerr << __FILE__ << " (" << __LINE__ << ") " << #x << " == " << (x) << std::endl

Funkcje

- void **dodaj** (std::shared_ptr< **wezel** > &pKorzen, const double wartosc)
- std::tuple< std::shared_ptr< **wezel** >, double, std::shared_ptr< **wezel** > > **podziel_wezel** (std::shared_ptr< **wezel** > &pWezel)
- void **podmien_wartosci** (std::shared_ptr< **wezel** > &pWezel, const double wartosc, std::shared_ptr< **wezel** > &pLewy, double mediana, std::shared_ptr< **wezel** > &pPrawy)
- void **gdy_za_duzo_w_wezle** (std::shared_ptr< **wezel** > &pWezel, double wartosc)
- std::shared_ptr< **wezel** > **utworz_nowy_korzen_drzewa** (std::shared_ptr< **wezel** > &pLewy, const double mediana, std::shared_ptr< **wezel** > &pPrawy)
- std::shared_ptr< **wezel** > **znajdz_adres_potomka** (const std::shared_ptr< **wezel** > pWezel, const double wartosc)
- bool **czy_za_duzo_w_wezle** (const std::shared_ptr< **wezel** > pWezel)
- void **wsortuj_potomka** (const std::shared_ptr< **wezel** > &potomek, std::list< std::shared_ptr< **wezel** >> &lista_potomkow)
- void **wsortuj_wartosc** (const double wartosc, std::list< double > &wartosci)
- void **wypisz** (const std::shared_ptr< **wezel** > &pKorzen, const int wciecie)
- void **zapisz_do_pliku** (std::ofstream &plik, const std::shared_ptr< **wezel** > &pKorzen, const int wciecie)

4.1.1 Dokumentacja definicji

4.1.1.1 debug

```
#define debug(
             x ) std::cerr << __FILE__ << " (" << __LINE__ << ") " << #x << " == " <<
(x) << std::endl
```

4.1.2 Dokumentacja funkcji

4.1.2.1 czy_za_duzo_w_wezle()

```
bool czy_za_duzo_w_wezle (
             const std::shared_ptr<  wezel > pWezel )
```

Funkcja sprawdza czy w wezle nie ma zbyt wielu wartosci

Parametry

pWezel	Wezel do sprawdzenia

Zwraca

true – za duzo wartosci w wezle, potrzebna modyfikacja, false – poprawna ilosc wartosci w wezle

4.1.2.2 dodaj()

```
void dodaj (
             std::shared_ptr<  wezel > & pKorzen,
             const double wartosc )
```

Funkcja dodajaca wartosc do B-drzewa.

Parametry

in,out	*pKorzen*	Korzen drzewa (jezeli drzewo jest puste, ma wartosc nullptr)
	wartosc	Wartosc do dodania do drzewa

4.1.2.3 gdy_za_duzo_w_wezle()

```
void gdy_za_duzo_w_wezle (
            std::shared_ptr<  wezel > & pWezel,
            double wartosc )
```

Funkcja ktora stanowi posrednik miedzy funkcja dodaj, a funkcjami ktore modyfikuja wezly

Parametry

in,out	*pWezel*	Wezel drzewa
	wartosc	Wartosc ktora spowodowala przeciazenie wezla

4.1.2.4 podmien_wartosci()

```
void podmien_wartosci (
            std::shared_ptr<  wezel > & pWezel,
            const double wartosc,
            std::shared_ptr<  wezel > & pLewy,
            double mediana,
            std::shared_ptr<  wezel > & pPrawy )
```

Funkcja usuwa zbyt duzy wezel i wklada na jego miejsce 2 nowe wezly (lewy i prawy) oraz mediane do rodzica

Parametry

in,out	*pWezel*	Rozbity wezel
	wartosc	Wartosc ktora wywolala rozbicie
in,out	*pLewy*	Lewy wezel utworzony po rozbiciu oryginalnego wezla
	mediana	Mediana, czyli wartosc srodkowa, ktora idzie to rodzica wezla
in,out	*pPrawy*	Prawy wezel utworzony po rozbiciu oryginalnego wezla

4.1.2.5 podziel_wezel()

```
std::tuple<std::shared_ptr< wezel>, double, std::shared_ptr< wezel> > podziel_wezel (
            std::shared_ptr<  wezel > & pWezel )
```

Funkcja dzieli zbyt duzy wezel na prawa i lewa wartosc oraz mediane i przekazuje do funkcji gdy_za_duzo_w_wezle, ktora decyduje co robic dalej

Parametry

in,out	*pWezel*	Wezel drzewa do podzielenia

Zwraca

Lewy wezel

Mediana

Prawy wezel

4.1.2.6 utworz_nowy_korzen_drzewa()

```
std::shared_ptr< wezel> utworz_nowy_korzen_drzewa (
            std::shared_ptr<  wezel > & pLewy,
            const double mediana,
            std::shared_ptr<  wezel > & pPrawy )
```

Funkcja przyjmuje lewy i prawy wezel oraz mediane (po rozbiciu starego korzenia), tworzy nowy korzen, dodaje do niego lewe i prawe potomki, zwraca wskaznik na nowy korzen, ktory podmieniamy z oryginalnym wskaznikiem

Parametry

in,out	*pLewy*	Lewy wezel
	mediana	Mediana
in,out	*pPrawy*	Prawy wezel

Zwraca

Nowy korzen

4.1.2.7 wsortuj_potomka()

```
void wsortuj_potomka (
            const std::shared_ptr<  wezel > & potomek,
            std::list< std::shared_ptr<  wezel >> & lista_potomkow )
```

Funkcja dodaje potomka do listy potomkow, wsadzajac go w odpowiednie miejsce, tak by lista byla posortowana rosnaco wg pierwszych wartosci potomkow

Parametry

in,out	*potomek*	Potomek do wsortowania
in,out	*lista_potomkow*	Lista do ktorej wsortujemy potomka

4.1.2.8 wsortuj_wartosc()

```
void wsortuj_wartosc (
```

```
            const double wartosc,
            std::list< double > & wartosci )
```

Funkcja dodaje wartosc do listy i ja sortuje

Parametry

	wartosc	Dodawana wartosc
in,out	*wartosci*	Lista wartosci do ktorej dodajemy

4.1.2.9 wypisz()

```
void wypisz (
            const std::shared_ptr<  wezel > & pKorzen,
            const int wciecie )
```

Funkcja wypisujaca graf drzewa na standardowe wyjscie

Parametry

in,out	*pKorzen*	Korzen drzewa
	wciecie	Wspolczynnik wciecia

4.1.2.10 zapisz_do_pliku()

```
void zapisz_do_pliku (
            std::ofstream & plik,
            const std::shared_ptr<  wezel > & pKorzen,
            const int wciecie )
```

Funkcja zapisujaca graf drzewa do pliku

Parametry

in,out	*plik*	Strumien wyjsciowy do pliku
in,out	*pKorzen*	Korzen drzewa
	wciecie	Wspolczynnik wciecia

4.1.2.11 znajdz_adres_potomka()

```
std::shared_ptr< wezel> znajdz_adres_potomka (
            const std::shared_ptr<  wezel > pWezel,
            const double wartosc )
```

Funkcja znajduje adres potomka do ktorego kierujemy wartosc

Parametry

pWezel	Rodzic, wsrod ktorego potomkow szukamy takiego, do ktorego ma trafic wartosc
wartosc	Wartosc dla ktorej szukamy potomka

Zwraca

Szukany potomek

4.2 Dokumentacja pliku drzewo.h

```
#include <vector>
#include <string>
#include <memory>
#include <list>
```

Komponenty

- struct **wezel**

  Struktura wezla drzewa.

Funkcje

- void **dodaj** (std::shared_ptr< **wezel** > &pKorzen, const double wartosc)
- void **wypisz** (const std::shared_ptr< **wezel** > &pKorzen, const int wciecie)
- void **zapisz_do_pliku** (std::ofstream &plik, const std::shared_ptr< **wezel** > &pKorzen, const int wciecie)
- void **wsortuj_wartosc** (const double wartosc, std::list< double > &wartosci)
- bool **czy_za_duzo_w_wezle** (const std::shared_ptr< **wezel** > pWezel)
- void **gdy_za_duzo_w_wezle** (std::shared_ptr< **wezel** > &pWezel, double wartosc)
- std::tuple< std::shared_ptr< **wezel** >, double, std::shared_ptr< **wezel** > > **podziel_wezel** (std::shared_ptr< **wezel** > &pWezel)
- void **podmien_wartosci** (std::shared_ptr< **wezel** > &pWezel, const double wartosc, std::shared_ptr< **wezel** > &pLewy, double mediana, std::shared_ptr< **wezel** > &pPrawy)
- std::shared_ptr< **wezel** > **utworz_nowy_korzen_drzewa** (std::shared_ptr< **wezel** > &pLewy, const double mediana, std::shared_ptr< **wezel** > &pPrawy)
- std::shared_ptr< **wezel** > **znajdz_adres_potomka** (const std::shared_ptr< **wezel** > pWezel, const double wartosc)
- void **wsortuj_potomka** (const std::shared_ptr< **wezel** > &potomek, std::list< std::shared_ptr< **wezel** >> &lista_potomkow)

4.2.1 Dokumentacja funkcji

4.2.1.1 czy_za_duzo_w_wezle()

```
bool czy_za_duzo_w_wezle (
            const std::shared_ptr< wezel > pWezel )
```

Funkcja sprawdza czy w wezle nie ma zbyt wielu wartosci

Parametry

pWezel	Wezel do sprawdzenia

Zwraca

true – za duzo wartosci w wezle, potrzebna modyfikacja, false – poprawna ilosc wartosci w wezle

4.2.1.2 dodaj()

```
void dodaj (
            std::shared_ptr< wezel > & pKorzen,
            const double wartosc )
```

Funkcja dodajaca wartosc do B-drzewa.

Parametry

in,out	*pKorzen*	Korzen drzewa (jezeli drzewo jest puste, ma wartosc nullptr)
	wartosc	Wartosc do dodania do drzewa

4.2.1.3 gdy_za_duzo_w_wezle()

```
void gdy_za_duzo_w_wezle (
            std::shared_ptr< wezel > & pWezel,
            double wartosc )
```

Funkcja ktora stanowi posrednik miedzy funkcja dodaj, a funkcjami ktore modyfikuja wezly

Parametry

in,out	*pWezel*	Wezel drzewa
	wartosc	Wartosc ktora spowodowala przeciazenie wezla

4.2.1.4 podmien_wartosci()

```
void podmien_wartosci (
            std::shared_ptr< wezel > & pWezel,
            const double wartosc,
            std::shared_ptr< wezel > & pLewy,
```

```
            double mediana,
            std::shared_ptr<  wezel > & pPrawy )
```

Funkcja usuwa zbyt duzy wezel i wklada na jego miejsce 2 nowe wezly (lewy i prawy) oraz mediane do rodzica

Parametry

in,out	*pWezel*	Rozbity wezel
	wartosc	Wartosc ktora wywolala rozbicie
in,out	*pLewy*	Lewy wezel utworzony po rozbiciu oryginalnego wezla
	mediana	Mediana, czyli wartosc srodkowa, ktora idzie to rodzica wezla
in,out	*pPrawy*	Prawy wezel utworzony po rozbiciu oryginalnego wezla

4.2.1.5 podziel_wezel()

```
std::tuple<std::shared_ptr< wezel>, double, std::shared_ptr< wezel> > podziel_wezel (
            std::shared_ptr<  wezel > & pWezel )
```

Funkcja dzieli zbyt duzy wezel na prawa i lewa wartosc oraz mediane i przekazuje do funkcji gdy_za_duzo_w_wezle, ktora decyduje co robic dalej

Parametry

in,out	*pWezel*	Wezel drzewa do podzielenia

Zwraca

Lewy wezel

Mediana

Prawy wezel

4.2.1.6 utworz_nowy_korzen_drzewa()

```
std::shared_ptr< wezel> utworz_nowy_korzen_drzewa (
            std::shared_ptr<  wezel > & pLewy,
            const double mediana,
            std::shared_ptr<  wezel > & pPrawy )
```

Funkcja przyjmuje lewy i prawy wezel oraz mediane (po rozbiciu starego korzenia), tworzy nowy korzen, dodaje do niego lewe i prawe potomki, zwraca wskaznik na nowy korzen, ktory podmieniamy z oryginalnym wskaznikiem

Parametry

in,out	*pLewy*	Lewy wezel
	mediana	Mediana
in,out	*pPrawy*	Prawy wezel

Zwraca

Nowy korzen

4.2.1.7 wsortuj_potomka()

```
void wsortuj_potomka (
            const std::shared_ptr<  wezel > & potomek,
            std::list< std::shared_ptr<  wezel >> & lista_potomkow )
```

Funkcja dodaje potomka do listy potomkow, wsadzajac go w odpowiednie miejsce, tak by lista byla posortowana rosnaco wg pierwszych wartosci potomkow

Parametry

in,out	*potomek*	Potomek do wsortowania
in,out	*lista_potomkow*	Lista do ktorej wsortujemy potomka

4.2.1.8 wsortuj_wartosc()

```
void wsortuj_wartosc (
            const double wartosc,
            std::list< double > & wartosci )
```

Funkcja dodaje wartosc do listy i ja sortuje

Parametry

	wartosc	Dodawana wartosc
in,out	*wartosci*	Lista wartosci do ktorej dodajemy

4.2.1.9 wypisz()

```
void wypisz (
            const std::shared_ptr<  wezel > & pKorzen,
            const int wciecie )
```

Funkcja wypisujaca graf drzewa na standardowe wyjscie

Parametry

in,out	*pKorzen*	Korzen drzewa
	wciecie	Wspolczynnik wciecia

4.2.1.10 zapisz_do_pliku()

```
void zapisz_do_pliku (
            std::ofstream & plik,
            const std::shared_ptr<  wezel > & pKorzen,
            const int wciecie )
```

Funkcja zapisujaca graf drzewa do pliku

Parametry

in,out	*plik*	Strumien wyjsciowy do pliku
in,out	*pKorzen*	Korzen drzewa
	wciecie	Wspolczynnik wciecia

4.2.1.11 znajdz_adres_potomka()

```
std::shared_ptr< wezel> znajdz_adres_potomka (
            const std::shared_ptr<  wezel > pWezel,
            const double wartosc )
```

Funkcja znajduje adres potomka do ktorego kierujemy wartosc

Parametry

pWezel	Rodzic, wsrod ktorego potomkow szukamy takiego, do ktorego ma trafic wartosc
wartosc	Wartosc dla ktorej szukamy potomka

Zwraca

Szukany potomek

4.3 Dokumentacja pliku funkcje.cpp

```
#include <iostream>
#include <vector>
#include <sstream>
#include <fstream>
#include <iomanip>
#include <list>
#include <algorithm>
#include "funkcje.h"
#include "struktury.h"
#include "drzewo.h"
```

Definicje

- #define **debug**(x) std::cerr << __FILE__ << " (" << __LINE__ << ") " << #x << " == " << (x) << std::endl

Funkcje

- std::vector< **komenda** > **odczytaj_komendy_z_pliku** (const std::string &nazwa)
- std::string **to_string** (const **typ** &t)
- void **test_komend** (const std::vector< **komenda** > &ciag_komend)
- void **help** (const std::string &nazwa_pliku_wykonywalnego)
- bool **sprawdz_parametry_programu** (int argc, char *params[ ], std::string &nazwa_pliku)
- void **wykonaj_komendy** (std::shared_ptr< **wezel** > &pKorzen, const std::vector< **komenda** > &ciag_komend)

4.3.1 Dokumentacja definicji

4.3.1.1 debug

```
#define debug(
              x ) std::cerr << __FILE__ << " (" << __LINE__ << ") " << #x << " == " <<
(x) << std::endl
```

4.3.2 Dokumentacja funkcji

4.3.2.1 help()

```
void help (
             const std::string & nazwa_pliku_wykonywalnego )
```

Funkcja wyswietlajaca notke przypominajaca jak uruchomic program

Parametry

in,out	*nazwa_pliku_wykonywalnego*	Nazwa pliku wykonywalnego

4.3.2.2 odczytaj_komendy_z_pliku()

```
std::vector< komenda> odczytaj_komendy_z_pliku (
             const std::string & nazwa )
```

Funkcja odczytuje z pliku wejsciowego komendy wraz z parametrami, pomijajac komentarze

Parametry

in,out	*Nazwa*	strumienia pliku wejsciowego

Zwraca

Ciag komend

4.3.2.3 sprawdz_parametry_programu()

```
bool sprawdz_parametry_programu (
            int argc,
            char * params[],
            std::string & nazwa_pliku )
```

Parametry

	argc	Pierwszy parametr main
	params	Drugi parametr main
in,out	*nazwa_pliku*	Jezeli poprawne parametry wywolania programu, to tutaj bedzie odczytana nazwa pliku.

Zwraca

true – wszystko OK, parametry poprawne, false – parametry bledne

4.3.2.4 test_komend()

```
void test_komend (
            const std::vector< komenda > & ciag_komend )
```

Testowe wypisanie na ekran nazwy kazdej komendy (i jej parametrow)

Parametry

in,out	*ciag_komend*	Komendy zebrane z pliku wejsciowego

4.3.2.5 to_string()

```
std::string to_string (
```

```
            const  typ & t )
```

4.3.2.6 wykonaj_komendy()

```
void wykonaj_komendy (
            std::shared_ptr<  wezel > & pKorzen,
            const std::vector<  komenda > & ciag_komend )
```

Funkcja wykonujaca komendy zebrane z pliku wejsciowego

Parametry

in,out	*pKorzen*	Korzen drzewa
in,out	*ciag_komend*	Komendy zebrane z pliku wejsciowego

4.4 Dokumentacja pliku funkcje.h

```
#include <vector>
#include <string>
#include <memory>
#include "struktury.h"
#include "drzewo.h"
```

Funkcje

- std::vector< **komenda** > **odczytaj_komendy_z_pliku** (const std::string &nazwa)
- void **test_komend** (const std::vector< **komenda** > &ciag_komend)
- bool **sprawdz_parametry_programu** (int argc, char ∗params[ ], std::string &nazwa_pliku)
- void **help** (const std::string &nazwa_pliku_wykonywalnego)
- void **wykonaj_komendy** (std::shared_ptr< **wezel** > &pKorzen, const std::vector< **komenda** > &ciag_↩ komend)

4.4.1 Dokumentacja funkcji

4.4.1.1 help()

```
void help (
            const std::string & nazwa_pliku_wykonywalnego )
```

Funkcja wyswietlajaca notke przypominajaca jak uruchomic program

Parametry

in,out	*nazwa_pliku_wykonywalnego*	Nazwa pliku wykonywalnego

4.4.1.2 odczytaj_komendy_z_pliku()

```
std::vector< komenda> odczytaj_komendy_z_pliku (
            const std::string & nazwa )
```

Funkcja odczytuje z pliku wejsciowego komendy wraz z parametrami, pomijajac komentarze

Parametry

in,out	*Nazwa*	strumienia pliku wejsciowego

Zwraca

Ciag komend

4.4.1.3 sprawdz_parametry_programu()

```
bool sprawdz_parametry_programu (
            int argc,
            char * params[],
            std::string & nazwa_pliku )
```

Parametry

	argc	Pierwszy parametr main
	params	Drugi parametr main
in,out	*nazwa_pliku*	Jezeli poprawne parametry wywolania programu, to tutaj bedzie odczytana nazwa pliku.

Zwraca

true – wszystko OK, parametry poprawne, false – parametry bledne

4.4.1.4 test_komend()

```
void test_komend (
            const std::vector<  komenda > & ciag_komend )
```

Testowe wypisanie na ekran nazwy kazdej komendy (i jej parametrow)

Parametry

in,out	*ciag_komend*	Komendy zebrane z pliku wejsciowego

4.4.1.5 wykonaj_komendy()

```
void wykonaj_komendy (
            std::shared_ptr< wezel > & pKorzen,
            const std::vector< komenda > & ciag_komend )
```

Funkcja wykonujaca komendy zebrane z pliku wejsciowego

Parametry

in,out	*pKorzen*	Korzen drzewa
in,out	*ciag_komend*	Komendy zebrane z pliku wejsciowego

4.5 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <memory>
#include <algorithm>
#include <numeric>
#include <random>
#include "funkcje.h"
#include "struktury.h"
#include "drzewo.h"
```

Definicje

- #define **debug**(x) std::cerr << __FILE__ << " (" << __LINE__ << ") " << #x << " == " << (x) << std::endl

Funkcje

- void **test** ()
- int **main** (int ile, char *params[ ])

4.5.1 Dokumentacja definicji

4.5.1.1 debug

```
#define debug(
             x ) std::cerr << __FILE__ << " (" << __LINE__ << ") " << #x << " == " <<
(x) << std::endl
```

4.5.2 Dokumentacja funkcji

4.5.2.1 main()

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

4.5.2.2 test()

```
void test ( )
```

4.6 Dokumentacja pliku struktury.h

```
#include <vector>
#include <string>
#include <memory>
#include <list>
```

Komponenty

- struct **komenda**
  Struktura komendy.

Wyliczenia

- enum class **typ** {
  rzad , **add** , **graph** , **print** ,
  append }
  Typ wyliczeniowy okreslajacy rodzaje komend.

4.6.1 Dokumentacja typów wyliczanych

4.6.1.1 typ

```
enum typ [strong]
```

Typ wyliczeniowy okreslajacy rodzaje komend.

Wartości wyliczeń

rzad	
add	Rzad drzewa.
graph	Komenda dodaj.
print	Komenda wypisz drzewo na standardowe urzadzenie wyjsciowe.
append	Komenda wypisz wartosci drzewa, lub zapisz graf do pliku. Inaczej print +, czyli dopisanie do pliku