ListaDwukierunkowa

Generated by Doxygen 1.12.0

1 Class Index 1
1.1 Class List . . . 1

2 File Index 1
2.1 File List . . . 1

3 Class Documentation 2
3.1 ListaDwukierunkowa Class Reference . . . 2
3.1.1 Detailed Description . . . 3
3.1.2 Constructor & Destructor Documentation . . . 3
3.1.3 Member Function Documentation . . . 3
3.1.4 Member Data Documentation . . . 6
3.2 Wezel Struct Reference . . . 6
3.2.1 Detailed Description . . . 7
3.2.2 Constructor & Destructor Documentation . . . 7
3.2.3 Member Data Documentation . . . 7

4 File Documentation 8
4.1 ListaDwukierunkowa/ListaDwukierunkowa.cpp File Reference . . . 8
4.2 ListaDwukierunkowa.cpp . . . 8
4.3 ListaDwukierunkowa/ListaDwukierunkowa.h File Reference . . . 10
4.4 ListaDwukierunkowa.h . . . 10
4.5 ListaDwukierunkowa/main.cpp File Reference . . . 11
4.5.1 Function Documentation . . . 11
4.6 main.cpp . . . 12

Index 13

1 Class Index

1.1 Class List

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

ListaDwukierunkowa
Klasa reprezentujaca liste dwukierunkowa 2

Wezel
Struktura reprezentujaca wezel w liscie dwukierunkowej 6

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

ListaDwukierunkowa/ListaDwukierunkowa.cpp **8**

ListaDwukierunkowa/ListaDwukierunkowa.h **10**

ListaDwukierunkowa/main.cpp **11**

3 Class Documentation

3.1 ListaDwukierunkowa Class Reference

Klasa reprezentujaca liste dwukierunkowa.

`#include <ListaDwukierunkowa.h>`

Public Member Functions

- ListaDwukierunkowa ()
  Konstruktor domyslny inicjalizujacy pusta liste.
- ~ListaDwukierunkowa ()
  Destruktor listy dwukierunkowej.
- void Dodawanie_napoczatek (int wartosc)
  Dodaje nowy element na poczatek listy.
- void Dodawanie_nakoniec (int wartosc)
  Dodaje nowy element na koniec listy.
- void Wyswietlanie_od_poczatku ()
  Wyswietla wszystkie elementy listy od poczatku.
- void Wyswietlanie_od_konca ()
  Wyswietla wszystkie elementy listy od konca.
- void Dodawanie_na_index (int wartosc, int index)
  Dodaje nowy element na okreslony indeks w liscie.
- void usuwanie_z_poczatku ()
  Usuwa pierwszy element z listy.
- void usuwanie_z_konca ()
  Usuwa ostatni element z listy.
- void usuwanie_z_indexu (int index)
  Usuwa element z listy na danym indeksie.
- void czyszczenie_listy ()
  Usuwa wszystkie elementy z listy.
- void wyswietl_nastepny ()
  Wyswietla nastepny element w liscie.
- void wyswietl_poprzedni ()
  Wyswietla poprzedni element w liscie.

Public Attributes

- Wezel * wezel

Private Attributes

- Wezel * poczatek
- Wezel * koniec
- int ilosc_elementow

3.1.1 Detailed Description

Klasa reprezentujaca liste dwukierunkowa.

Definition at line 26 of file ListaDwukierunkowa.h.

3.1.2 Constructor & Destructor Documentation

ListaDwukierunkowa()

```
ListaDwukierunkowa::ListaDwukierunkowa ()  [inline]
```

Konstruktor domyslny inicjalizujacy pusta liste.

Definition at line 38 of file ListaDwukierunkowa.h.

~ListaDwukierunkowa()

```
ListaDwukierunkowa::~ListaDwukierunkowa ()  [inline]
```

Destruktor listy dwukierunkowej.

Definition at line 43 of file ListaDwukierunkowa.h.

3.1.3 Member Function Documentation

czyszczenie_listy()

```
void ListaDwukierunkowa::czyszczenie_listy ()
```

Usuwa wszystkie elementy z listy.

Definition at line 187 of file ListaDwukierunkowa.cpp.

Dodawanie_na_index()

```
void ListaDwukierunkowa::Dodawanie_na_index (
            int wartosc,
            int index)
```

Dodaje nowy element na okreslony indeks w liscie.

Parameters

wartosc	Wartosc do dodania.
index	Indeks, na ktory ma zostac dodany element.

Definition at line 87 of file ListaDwukierunkowa.cpp.

Dodawanie_nakoniec()

```
void ListaDwukierunkowa::Dodawanie_nakoniec (
            int wartosc)
```

Dodaje nowy element na koniec listy.

Parameters

wartosc	Wartosc do dodania na koniec listy.

Definition at line 34 of file ListaDwukierunkowa.cpp.

Dodawanie_napoczatek()

```
void ListaDwukierunkowa::Dodawanie_napoczatek (
            int wartosc)
```

Dodaje nowy element na poczatek listy.

Parameters

wartosc	Wartosc do dodania na poczatek listy.

Definition at line 9 of file ListaDwukierunkowa.cpp.

usuwanie_z_indexu()

```
void ListaDwukierunkowa::usuwanie_z_indexu (
            int index)
```

Usuwa element z listy na danym indeksie.

Parameters

index	Indeks elementu do usuniecia.

Definition at line 161 of file ListaDwukierunkowa.cpp.

usuwanie_z_konca()

```
void ListaDwukierunkowa::usuwanie_z_konca ()
```

Usuwa ostatni element z listy.

Definition at line 137 of file ListaDwukierunkowa.cpp.

usuwanie_z_poczatku()

```
void ListaDwukierunkowa::usuwanie_z_poczatku ()
```

Usuwa pierwszy element z listy.

Definition at line 115 of file ListaDwukierunkowa.cpp.

wyswietl_nastepny()

```
void ListaDwukierunkowa::wyswietl_nastepny ()
```

Wyswietla nastepny element w liscie.

Definition at line 196 of file ListaDwukierunkowa.cpp.

wyswietl_poprzedni()

```
void ListaDwukierunkowa::wyswietl_poprzedni ()
```

Wyswietla poprzedni element w liscie.

Definition at line 210 of file ListaDwukierunkowa.cpp.

Wyswietlanie_od_konca()

```
void ListaDwukierunkowa::Wyswietlanie_od_konca ()
```

Wyswietla wszystkie elementy listy od konca.

Definition at line 73 of file ListaDwukierunkowa.cpp.

Wyswietlanie_od_poczatku()

```
void ListaDwukierunkowa::Wyswietlanie_od_poczatku ()
```

Wyswietla wszystkie elementy listy od poczatku.

Definition at line 57 of file ListaDwukierunkowa.cpp.

3.1.4 Member Data Documentation

ilosc_elementow

```
int ListaDwukierunkowa::ilosc_elementow  [private]
```

Liczba elementow w liscie.

Definition at line 30 of file ListaDwukierunkowa.h.

koniec

```
Wezel* ListaDwukierunkowa::koniec  [private]
```

Wskaznik na koniec listy.

Definition at line 29 of file ListaDwukierunkowa.h.

poczatek

```
Wezel* ListaDwukierunkowa::poczatek  [private]
```

Wskaznik na poczatek listy.

Definition at line 28 of file ListaDwukierunkowa.h.

wezel

```
Wezel* ListaDwukierunkowa::wezel
```

Wskaznik na biezacy wezel w liscie.

Definition at line 33 of file ListaDwukierunkowa.h.

The documentation for this class was generated from the following files:

- ListaDwukierunkowa/ListaDwukierunkowa.h
- ListaDwukierunkowa/ListaDwukierunkowa.cpp

3.2 Wezel Struct Reference

Struktura reprezentujaca wezel w liscie dwukierunkowej.

```
#include <ListaDwukierunkowa.h>
```

Public Member Functions

- Wezel ()

  Konstruktor domyslny inicjalizujacy wartosc oraz wskazniki.
- ~Wezel ()

  Destruktor wezla.

Public Attributes

- int wartosc
- Wezel * nastepny_wezel
- Wezel * poprzedni_wezel

3.2.1 Detailed Description

Struktura reprezentujaca wezel w liscie dwukierunkowej.

Definition at line 7 of file ListaDwukierunkowa.h.

3.2.2 Constructor & Destructor Documentation

Wezel()

```
Wezel::Wezel () [inline]
```

Konstruktor domyslny inicjalizujacy wartosc oraz wskazniki.

Definition at line 15 of file ListaDwukierunkowa.h.

~Wezel()

```
Wezel::~Wezel () [inline]
```

Destruktor wezla.

Definition at line 20 of file ListaDwukierunkowa.h.

3.2.3 Member Data Documentation

nastepny_wezel

```
Wezel* Wezel::nastepny_wezel
```

Wskaznik na nastepny wezel w liscie.

Definition at line 9 of file ListaDwukierunkowa.h.

poprzedni_wezel

```
Wezel* Wezel::poprzedni_wezel
```

Wskaznik na poprzedni wezel w liscie.

Definition at line 10 of file ListaDwukierunkowa.h.

wartosc

```
int Wezel::wartosc
```

Wartosc przechowywana w wezle.

Definition at line 8 of file ListaDwukierunkowa.h.

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

- ListaDwukierunkowa/ListaDwukierunkowa.h

4 File Documentation

4.1 ListaDwukierunkowa/ListaDwukierunkowa.cpp File Reference

```
#include <iostream>
#include "ListaDwukierunkowa.h"
```

4.2 ListaDwukierunkowa.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include "ListaDwukierunkowa.h"
00003
00009 void ListaDwukierunkowa::Dodawanie_napoczatek(int wartosc) {
00010     if (ilosc_elementow > 0) {
00011         Wezel* nowyWezel = new Wezel;
00012         poczatek->poprzedni_wezel = nowyWezel;
00013         poczatek->poprzedni_wezel->nastepny_wezel = poczatek;
00014         poczatek = nowyWezel;
00015         nowyWezel->wartosc = wartosc;
00016     }
00017     else {
00018         Wezel* nowyWezel = new Wezel;
00019         poczatek = nowyWezel;
00020         koniec = nowyWezel;
00021         nowyWezel->wartosc = wartosc;
00022     }
00023     if (wezel == nullptr) {
00024         wezel = poczatek;
00025     }
00026     ilosc_elementow++;
00027 }
00028
00034 void ListaDwukierunkowa::Dodawanie_nakoniec(int wartosc) {
00035     if (ilosc_elementow > 0) {
00036         Wezel* nowyWezel = new Wezel;
00037         koniec->nastepny_wezel = nowyWezel;
00038         koniec->nastepny_wezel->poprzedni_wezel = koniec;
00039         koniec = nowyWezel;
```

```
00040         nowyWezel->wartosc = wartosc;
00041     }
00042     else {
00043         Wezel* nowyWezel = new Wezel;
00044         poczatek = nowyWezel;
00045         koniec = nowyWezel;
00046         nowyWezel->wartosc = wartosc;
00047     }
00048     if (wezel == nullptr) {
00049         wezel = poczatek;
00050     }
00051     ilosc_elementow++;
00052 }
00053
00057 void ListaDwukierunkowa::Wyswietlanie_od_poczatku() {
00058     Wezel* wskaznik = poczatek;
00059     for (int i = 0; i < ilosc_elementow; i++) {
00060         std::cout << wskaznik->wartosc << std::endl;
00061         if (wskaznik->nastepny_wezel != nullptr) {
00062             wskaznik = wskaznik->nastepny_wezel;
00063         }
00064         else {
00065             return;
00066         }
00067     }
00068 }
00069
00073 void ListaDwukierunkowa::Wyswietlanie_od_konca() {
00074     Wezel* wskaznik = koniec;
00075     for (int i = 0; i < ilosc_elementow; i++) {
00076         std::cout << wskaznik->wartosc << std::endl;
00077         wskaznik = wskaznik->poprzedni_wezel;
00078     }
00079 }
00080
00087 void ListaDwukierunkowa::Dodawanie_na_index(int wartosc, int index) {
00088     if (index > 0 && index < ilosc_elementow) {
00089         Wezel* nowyWezel = new Wezel;
00090         Wezel* wskaznik = poczatek;
00091         for (int i = 0; i < index - 1; i++) {
00092             wskaznik = wskaznik->nastepny_wezel;
00093         }
00094         nowyWezel->nastepny_wezel = wskaznik->nastepny_wezel;
00095         wskaznik->nastepny_wezel->poprzedni_wezel = nowyWezel;
00096         wskaznik->nastepny_wezel = nowyWezel;
00097         nowyWezel->poprzedni_wezel = wskaznik;
00098         nowyWezel->wartosc = wartosc;
00099         ilosc_elementow++;
00100     }
00101     else if (index == 0) {
00102         Dodawanie_napoczatek(wartosc);
00103     }
00104     else if (index == ilosc_elementow) {
00105         Dodawanie_nakoniec(wartosc);
00106     }
00107     else {
00108         std::cout << "Index poza lista";
00109     }
00110 }
00111
00115 void ListaDwukierunkowa::usuwanie_z_poczatku() {
00116     if (!poczatek) {
00117         std::cout << "brak elementow\n";
00118     }
00119     else {
00120         if (ilosc_elementow > 1) {
00121             Wezel* obecny = poczatek->nastepny_wezel;
00122             delete obecny->poprzedni_wezel;
00123             poczatek = obecny;
00124         }
00125         else if (ilosc_elementow == 1) {
00126             delete poczatek;
00127             koniec = nullptr;
00128             poczatek = nullptr;
00129         }
00130     }
00131     ilosc_elementow--;
00132 }
00133
00137 void ListaDwukierunkowa::usuwanie_z_konca() {
00138     if (!koniec) {
00139         std::cout << "brak elementow\n";
00140     }
00141     else {
00142         if (ilosc_elementow > 1) {
00143             Wezel* obecny = koniec->poprzedni_wezel;
00144             delete obecny->nastepny_wezel;
```

```
00145             koniec = obecny;
00146         }
00147         else if (ilosc_elementow == 1) {
00148             delete koniec;
00149             koniec = nullptr;
00150             poczatek = nullptr;
00151         }
00152     }
00153     ilosc_elementow--;
00154 }
00155
00161 void ListaDwukierunkowa::usuwanie_z_indexu(int index) {
00162     if (index > 0 && index < ilosc_elementow - 1) {
00163         Wezel* wskaznik = poczatek;
00164         for (int i = 0; i < index; i++) {
00165             wskaznik = wskaznik->nastepny_wezel;
00166         }
00167         Wezel* dodatkowy = wskaznik->poprzedni_wezel;
00168         dodatkowy->nastepny_wezel = wskaznik->nastepny_wezel;
00169         wskaznik->nastepny_wezel->poprzedni_wezel = dodatkowy;
00170         delete wskaznik;
00171     }
00172     else if (index == 0) {
00173         usuwanie_z_poczatku();
00174     }
00175     else if (index == ilosc_elementow) {
00176         usuwanie_z_konca();
00177     }
00178     else {
00179         std::cout « "Index poza lista";
00180     }
00181     ilosc_elementow--;
00182 }
00183
00187 void ListaDwukierunkowa::czyszczenie_listy() {
00188     do {
00189         usuwanie_z_konca();
00190     } while (ilosc_elementow > 0);
00191 }
00192
00196 void ListaDwukierunkowa::wyswietl_nastepny() {
00197     if (wezel->nastepny_wezel != nullptr && wezel != nullptr) {
00198         wezel = wezel->nastepny_wezel;
00199     }
00200     else {
00201         std::cout « "nie ma nastepnego elementu listy";
00202         return;
00203     }
00204     std::cout « wezel->wartosc;
00205 }
00206
00210 void ListaDwukierunkowa::wyswietl_poprzedni() {
00211     if (wezel->poprzedni_wezel != nullptr) {
00212         wezel = wezel->poprzedni_wezel;
00213     }
00214     else {
00215         std::cout « "Nie ma poprzedniego elementu listy";
00216         return;
00217     }
00218     std::cout « wezel->wartosc;
00219 }
```

4.3 ListaDwukierunkowa/ListaDwukierunkowa.h File Reference

Classes

- struct Wezel

  Struktura reprezentujaca wezel w liscie dwukierunkowej.
- class ListaDwukierunkowa

  Klasa reprezentujaca liste dwukierunkowa.

4.4 ListaDwukierunkowa.h

Go to the documentation of this file.

```
00001 #pragma once
00002 using namespace std;
00003
00007 struct Wezel {
00008     int wartosc;
00009     Wezel* nastepny_wezel;
00010     Wezel* poprzedni_wezel;
00015     Wezel() : wartosc(NULL), poprzedni_wezel(nullptr), nastepny_wezel(nullptr) {}
00016
00020     ~Wezel() {}
00021 };
00022
00026 class ListaDwukierunkowa {
00027 private:
00028     Wezel* poczatek;
00029     Wezel* koniec;
00030     int ilosc_elementow;
00032 public:
00033     Wezel* wezel;
00038     ListaDwukierunkowa() : poczatek(nullptr), koniec(nullptr), ilosc_elementow(0), wezel(nullptr) {}
00039
00043     ~ListaDwukierunkowa() {}
00044
00050     void Dodawanie_napoczatek(int wartosc);
00051
00057     void Dodawanie_nakoniec(int wartosc);
00058
00062     void Wyswietlanie_od_poczatku();
00063
00067     void Wyswietlanie_od_konca();
00068
00075     void Dodawanie_na_index(int wartosc, int index);
00076
00080     void usuwanie_z_poczatku();
00081
00085     void usuwanie_z_konca();
00086
00092     void usuwanie_z_indexu(int index);
00093
00097     void czyszczenie_listy();
00098
00102     void wyswietl_nastepny();
00103
00107     void wyswietl_poprzedni();
00108 };
```

4.5 ListaDwukierunkowa/main.cpp File Reference

```
#include <iostream>
#include "ListaDwukierunkowa.h"
```

Functions

- int main ()

4.5.1 Function Documentation

main()

```
int main ()
```

Definition at line 6 of file main.cpp.

4.6 main.cpp

Go to the documentation of this file.

```
00001
00002 #include <iostream>
00003 #include "ListaDwukierunkowa.h"
00004
00005 using namespace std;
00006 int main()
00007 {
00008
00009
00010
00011     ListaDwukierunkowa a;
00012     a.Dodawanie_napoczatek(5);
00013     a.Dodawanie_nakoniec(6);
00014     a.Dodawanie_nakoniec(7);
00015     a.Dodawanie_nakoniec(9);
00016     /*a.Wyswietlanie_od_poczatku();
00017     cout « endl;*/
00018    /* a.Dodawanie_na_index(5,0);*/
00019     a.Wyswietlanie_od_poczatku();
00020     /*a.usuwanie_z_poczatku();*/
00021    /* a.usuwanie_z_indexu(3);*/
00022   /*  cout « endl;
00023     a.czyszczenie_listy();
00024     a.Wyswietlanie_od_poczatku();
00025    */
00026     a.wyswietl_nastepny();
00027     a.wyswietl_nastepny();
00028     a.wyswietl_nastepny();
00029     a.wyswietl_nastepny();
00030
00031 }
00032
```

Index

~ListaDwukierunkowa
ListaDwukierunkowa, 3
~Wezel
Wezel, 7

czyszczenie_listy
ListaDwukierunkowa, 3

Dodawanie_na_index
ListaDwukierunkowa, 3
Dodawanie_nakoniec
ListaDwukierunkowa, 4
Dodawanie_napoczatek
ListaDwukierunkowa, 4

ilosc_elementow
ListaDwukierunkowa, 6

koniec
ListaDwukierunkowa, 6

ListaDwukierunkowa, 2
~ListaDwukierunkowa, 3
czyszczenie_listy, 3
Dodawanie_na_index, 3
Dodawanie_nakoniec, 4
Dodawanie_napoczatek, 4
ilosc_elementow, 6
koniec, 6
ListaDwukierunkowa, 3
poczatek, 6
usuwanie_z_indexu, 4
usuwanie_z_konca, 4
usuwanie_z_poczatku, 5
wezel, 6
wyswietl_nastepny, 5
wyswietl_poprzedni, 5
Wyswietlanie_od_konca, 5
Wyswietlanie_od_poczatku, 5
ListaDwukierunkowa/ListaDwukierunkowa.cpp, 8
ListaDwukierunkowa/ListaDwukierunkowa.h, 10
ListaDwukierunkowa/main.cpp, 11, 12

main
main.cpp, 11
main.cpp
main, 11

nastepny_wezel
Wezel, 7

poczatek
ListaDwukierunkowa, 6
poprzedni_wezel
Wezel, 7

usuwanie_z_indexu
ListaDwukierunkowa, 4
usuwanie_z_konca
ListaDwukierunkowa, 4
usuwanie_z_poczatku
ListaDwukierunkowa, 5

wartosc
Wezel, 8
Wezel, 6
~Wezel, 7
nastepny_wezel, 7
poprzedni_wezel, 7
wartosc, 8
Wezel, 7
wezel
ListaDwukierunkowa, 6
wyswietl_nastepny
ListaDwukierunkowa, 5
wyswietl_poprzedni
ListaDwukierunkowa, 5
Wyswietlanie_od_konca
ListaDwukierunkowa, 5
Wyswietlanie_od_poczatku
ListaDwukierunkowa, 5