Lista dwukierunkowa

Generated by Doxygen 1.12.0

1 Class Index

1.1 Class List

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

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

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 czyszczacy pamiec przy usuwaniu listy.
- void wyswietl_od_poczatku ()
  Wyswietla elementy listy od poczatku.
- void wyswietl_od_konca ()
  Wyswietla elementy listy od konca.
- Wezel * dodaj_na_poczatek (int wartosc)
  Dodaje nowy element na poczatek listy.
- Wezel * dodaj_na_koniec (int wartosc)
  Dodaje nowy element na koniec listy.
- Wezel * dodaj_na_index (int wartosc, int index)
  Dodaje nowy element na podanym indeksie.
- void usun_z_poczatku ()
  Usuwa pierwszy element z listy.
- void usun_z_konca ()
  Usuwa ostatni element z listy.
- void usun_z_indexu (int index)
  Usuwa element na podanym indeksie.
- string wezel (int index)
  Zwraca wartosc wezla na podanym indeksie jako string.
- void Widok ()
  Interfejs konsolowy do zarzadzania lista.

Private Attributes

- Wezel * poczatek
- Wezel * koniec
- int ilosc

3.1.1 Detailed Description

Klasa reprezentujaca liste dwukierunkowa.

Zapewnia metody do dodawania, usuwania i wyswietlania elementow listy.

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 36 of file ListaDwukierunkowa.h.

~ListaDwukierunkowa()

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

Destruktor czyszczacy pamiec przy usuwaniu listy.

Definition at line 41 of file ListaDwukierunkowa.h.

3.1.3 Member Function Documentation

dodaj_na_index()

```
Wezel * ListaDwukierunkowa::dodaj_na_index (
            int wartosc,
            int index)
```

Dodaje nowy element na podanym indeksie.

Parameters

wartosc	Wartosc do dodania.
index	Indeks, na ktorym nalezy dodac nowy element.

Returns

Wskaznik do nowo dodanego wezla, lub nullptr jesli indeks jest nieprawidlowy.

Definition at line 105 of file ListaDwukierunkowa.cpp.

dodaj_na_koniec()

```
Wezel * ListaDwukierunkowa::dodaj_na_koniec (
            int wartosc)
```

Dodaje nowy element na koniec listy.

Parameters

wartosc	Wartosc do dodania na koniec listy.

Returns

Wskaznik do nowo dodanego wezla.

Definition at line 78 of file ListaDwukierunkowa.cpp.

dodaj_na_poczatek()

```
Wezel * ListaDwukierunkowa::dodaj_na_poczatek (
            int wartosc)
```

Dodaje nowy element na poczatek listy.

Parameters

wartosc	Wartosc do dodania na poczatek listy.

Returns

Wskaznik do nowo dodanego wezla.

Definition at line 53 of file ListaDwukierunkowa.cpp.

usun_z_indexu()

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

Usuwa element na podanym indeksie.

Parameters

index	Indeks elementu do usuniecia.

Definition at line 178 of file ListaDwukierunkowa.cpp.

usun_z_konca()

```
void ListaDwukierunkowa::usun_z_konca ()
```

Usuwa ostatni element z listy.

Definition at line 153 of file ListaDwukierunkowa.cpp.

usun_z_poczatku()

```
void ListaDwukierunkowa::usun_z_poczatku ()
```

Usuwa pierwszy element z listy.

Definition at line 130 of file ListaDwukierunkowa.cpp.

wezel()

```
string ListaDwukierunkowa::wezel (
            int index)
```

Zwraca wartosc wezla na podanym indeksie jako string.

Parameters

index	Indeks wezla, ktorego wartosc ma zostac zwrocona.

Returns

Wartosc wezla jako string, lub pusty string jesli indeks jest nieprawidlowy.

Definition at line 202 of file ListaDwukierunkowa.cpp.

Widok()

```
void ListaDwukierunkowa::Widok ()
```

Interfejs konsolowy do zarzadzania lista.

Umozliwia uzytkownikowi wykonywanie operacji na liscie, takich jak dodawanie, usuwanie, czy wyswietlanie elementow.

Definition at line 224 of file ListaDwukierunkowa.cpp.

wyswietl_od_konca()

```
void ListaDwukierunkowa::wyswietl_od_konca ()
```

Wyswietla elementy listy od konca.

Wypisuje kazdy element listy zaczynajac od ostatniego elementu.

Definition at line 32 of file ListaDwukierunkowa.cpp.

wyswietl_od_poczatku()

```
void ListaDwukierunkowa::wyswietl_od_poczatku ()
```

Wyswietla elementy listy od poczatku.

Wypisuje kazdy element listy zaczynajac od pierwszego elementu.

Definition at line 12 of file ListaDwukierunkowa.cpp.

3.1.4 Member Data Documentation

ilosc

```
int ListaDwukierunkowa::ilosc  [private]
```

Liczba elementow w liscie.

Definition at line 30 of file ListaDwukierunkowa.h.

koniec

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

Wskaznik na ostatni element listy.

Definition at line 29 of file ListaDwukierunkowa.h.

poczatek

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

Wskaznik na pierwszy element listy.

Definition at line 28 of file ListaDwukierunkowa.h.

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

- E:/ProgramowanieZaawansowane/ZuchowiczDominik.ListaDwukierunkowa/Example/Example/ListaDwukierunkowa.h
- E:/ProgramowanieZaawansowane/ZuchowiczDominik.ListaDwukierunkowa/Example/Example/ListaDwukierunkowa.cpp

3.2 Wezel Struct Reference

Struktura reprezentujaca pojedynczy wezel listy dwukierunkowej.

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

Public Member Functions

- Wezel ()

  Konstruktor domyslny inicjalizujacy pola wezla.

Public Attributes

- int dane
- Wezel * nastepny
- Wezel * poprzedni

3.2.1 Detailed Description

Struktura reprezentujaca pojedynczy wezel listy dwukierunkowej.

Kazdy wezel zawiera dane oraz wskazniki na nastepny i poprzedni element listy.

Definition at line 10 of file ListaDwukierunkowa.h.

3.2.2 Constructor & Destructor Documentation

Wezel()

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

Konstruktor domyslny inicjalizujacy pola wezla.

Definition at line 18 of file ListaDwukierunkowa.h.

3.2.3 Member Data Documentation

dane

```
int Wezel::dane
```

Wartosc przechowywana w wezle.

Definition at line 11 of file ListaDwukierunkowa.h.

nastepny

```
Wezel* Wezel::nastepny
```

Wskaznik na nastepny element listy.

Definition at line 12 of file ListaDwukierunkowa.h.

poprzedni

```
Wezel* Wezel::poprzedni
```

Wskaznik na poprzedni element listy.

Definition at line 13 of file ListaDwukierunkowa.h.

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

- E:/ProgramowanieZaawansowane/ZuchowiczDominik.ListaDwukierunkowa/Example/Example/ListaDwukierunkowa.h

4 File Documentation

4.1 E:/ProgramowanieZaawansowane/ZuchowiczDominik.ListaDwukierunkowa/Example/Example/Example.cpp File Reference

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

Functions

- int main ()

4.1.1 Function Documentation

main()

```
int main ()
```

Definition at line 7 of file Example.cpp.

4.2 Example.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include <string>
00003 #include "ListaDwukierunkowa.h"
00004 using namespace std;
00005
00006
00007 int main()
00008 {
00009     ListaDwukierunkowa lista;
00010     lista.Widok();
00011
00012 }
```

4.3 E:/ProgramowanieZaawansowane/ZuchowiczDominik.ListaDwukierunkowa/↩ Example/Example/ListaDwukierunkowa.cpp File Reference

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

4.4 ListaDwukierunkowa.cpp

Go to the documentation of this file.

```
00001 #include "ListaDwukierunkowa.h"
00002 #include <iostream>
00003 #include <string>
00004
00005
00012     void ListaDwukierunkowa::wyswietl_od_poczatku() {
00013         if (!koniec) {}
00014         else {
00015             Wezel* wskaznik = poczatek;
00016             int i = 0;
00017             do {
00018                 if (wskaznik->nastepny != nullptr) cout « "(index: " « i « "; wartosc: " «
      wskaznik->dane « "), ";
00019                 else cout « "(index: " « i « "; wartosc: " « wskaznik->dane « ") ";
00020                 wskaznik = wskaznik->nastepny;
00021                 i++;
00022             } while (wskaznik != nullptr);
00023         }
00024     }
00025
00032     void ListaDwukierunkowa::wyswietl_od_konca() {
00033         if (!koniec) {}
00034         else {
00035             Wezel* wskaznik = koniec;
00036             int i = ilosc - 1;
00037             do {
00038                 if (wskaznik->poprzedni != nullptr) cout « "(index: " « i « "; wartosc: " «
      wskaznik->dane « "), ";
00039                 else cout « "(index: " « i « "; wartosc: " « wskaznik->dane « ") ";
00040                 wskaznik = wskaznik->poprzedni;
00041                 i--;
00042             } while (wskaznik != nullptr);
00043         }
00044     }
00045
00053     Wezel* ListaDwukierunkowa::dodaj_na_poczatek(int wartosc) {
00054         Wezel* nowyWezel = new Wezel;
00055         if (!poczatek) {
00056             poczatek = nowyWezel;
00057             koniec = nowyWezel;
00058             nowyWezel->dane = wartosc;
00059         }
00060         else
00061         {
00062             nowyWezel->nastepny = poczatek;
00063             poczatek->poprzedni = nowyWezel;
00064             poczatek = nowyWezel;
00065             nowyWezel->dane = wartosc;
00066         }
00067         ilosc++;
00068         return nowyWezel;
00069     }
00070
00078     Wezel* ListaDwukierunkowa::dodaj_na_koniec(int wartosc) {
00079         Wezel* nowyWezel = new Wezel;
00080         if (!koniec) {
00081             koniec = nowyWezel;
00082             poczatek = nowyWezel;
00083             nowyWezel->dane = wartosc;
00084         }
00085         else
00086         {
00087             nowyWezel->poprzedni = koniec;
00088             koniec->nastepny = nowyWezel;
00089             koniec = nowyWezel;
00090             nowyWezel->dane = wartosc;
00091         }
```

```
00092         ilosc++;
00093         return nowyWezel;
00094     }
00095
00096
00105     Wezel* ListaDwukierunkowa::dodaj_na_index(int wartosc, int index) {
00106         if (index > 0 && index < ilosc - 1) {
00107             Wezel* nowyWezel = new Wezel;
00108             Wezel* wskaznik = poczatek;
00109             for (int i = 0; i < index - 1; i++)
00110             {
00111                 wskaznik = wskaznik->nastepny;
00112             }
00113             nowyWezel->nastepny = wskaznik->nastepny;
00114             wskaznik->nastepny->poprzedni = nowyWezel;
00115             wskaznik->nastepny = nowyWezel;
00116             nowyWezel->poprzedni = wskaznik;
00117             nowyWezel->dane = wartosc;
00118             ilosc++;
00119             return nowyWezel;
00120         }
00121         else if (index == 0) dodaj_na_poczatek(wartosc);
00122         else if (index == ilosc - 1) dodaj_na_koniec(wartosc);
00123         else cout « "Index poza lista";
00124     }
00125
00130     void ListaDwukierunkowa::usun_z_poczatku() {
00131         if (ilosc == 0) {
00132             cout « "brak elementow\n";
00133         }
00134         else
00135             if (ilosc >= 2)
00136             {
00137                 Wezel* chwilowy = poczatek;
00138                 poczatek = poczatek->nastepny;
00139                 poczatek->poprzedni = nullptr;
00140                 delete chwilowy;
00141             }
00142             else {
00143                 delete poczatek;
00144                 koniec = nullptr;
00145                 poczatek = nullptr;
00146             }
00147
00148         ilosc--;
00149     }
00153     void ListaDwukierunkowa::usun_z_konca() {
00154         if (ilosc == 0) {
00155             cout « "brak elementow\n";
00156         }
00157         else if (ilosc > 1)
00158         {
00159             Wezel* chwilowy = koniec;
00160             koniec = koniec->poprzedni;
00161             koniec->nastepny = nullptr;
00162             delete chwilowy;
00163         }
00164         else if (ilosc == 1) {
00165             delete koniec;
00166             poczatek = nullptr;
00167             koniec = nullptr;
00168         }
00169         ilosc--;
00170     }
00171
00178     void ListaDwukierunkowa::usun_z_indexu(int index) {
00179         if (index > 0 && index < ilosc - 1) {
00180             Wezel* wskaznik = poczatek;
00181             for (int i = 0; i < index; i++) {
00182                 wskaznik = wskaznik->nastepny;
00183             }
00184             wskaznik->poprzedni->nastepny = wskaznik->nastepny;
00185             wskaznik->nastepny->poprzedni = wskaznik->poprzedni;
00186             delete wskaznik;
00187             ilosc--;
00188         }
00189         else if (index == 0) usun_z_poczatku();
00190         else if (index == ilosc - 1) usun_z_konca();
00191         else cout « "Index poza lista";
00192
00193     }
00194
00202     string ListaDwukierunkowa::wezel(int index) {
00203         if (poczatek == nullptr) return "";
00204         if (index == -1) return "";
00205         Wezel* wskaznik = poczatek;
00206         for (int i = 0; i < index; i++)
```

```
00207         {
00208             if (wskaznik->nastepny != nullptr) wskaznik = wskaznik->nastepny;
00209             else {
00210                 wskaznik = nullptr;
00211                 return "";
00212             }
00213         }
00214         if (wskaznik != nullptr) return to_string(wskaznik->dane);
00215 
00216     }
00217 
00224     void ListaDwukierunkowa::Widok() {
00225 
00226         int wybrany_index = 0, komenda = 0, wartosc = 0, wartosc2 = 0, widok = 0;
00227 
00228         do {
00229 
00230             system("cls");
00231 
00232             cout << "1 - poprzedni element\n2 - nastepny element\n3 - dodaj na poczatek\n4 - dodaj na
      koniec\n5 - dodaj na konkretne miejsce\n6 - usun z poczatku\n7 - usun z konca\n8 - usun z konkretnego
      indexu\n9 - wyswietl liste od poczatku\n10 - wyswietl liste od konca\n0 - wyjdz\n\n";
00233 
00234             if (widok == 0) {
00235                 if (wezel(wybrany_index - 1) != "") cout << wezel(wybrany_index - 1); else  cout << "#";
00236                 ;           cout << " -> " << wezel(wybrany_index) << " <- ";
00237                 if (wezel(wybrany_index + 1) != "") cout << wezel(wybrany_index + 1); else  cout << "#";
00238 
00239                 cout << " \nindex: " << wybrany_index << "\n\nkomenda: ";
00240             }
00241             else if (widok == 1) wyswietl_od_poczatku();
00242             else if (widok == 2) wyswietl_od_konca();
00243 
00244 
00245             cin >> komenda;
00246 
00247 
00248             switch (komenda)
00249             {
00250             case 1:
00251                 if (wybrany_index > 0) wybrany_index--;
00252                 break;
00253             case 2:
00254                 if (wybrany_index < ilosc - 1) wybrany_index++;;
00255                 break;
00256             case 3:
00257                 cout << "Podaj wartosc: ";
00258                 cin >> wartosc;
00259                 dodaj_na_poczatek(wartosc);
00260                 break;
00261             case 4:
00262                 cout << "Podaj wartosc: ";
00263                 cin >> wartosc;
00264                 dodaj_na_koniec(wartosc);
00265                 break;
00266             case 5:
00267                 cout << "Podaj index: ";
00268                 cin >> wartosc2;
00269 
00270                 cout << "Podaj wartosc: ";
00271                 cin >> wartosc;
00272 
00273                 dodaj_na_index(wartosc, wartosc2);
00274                 break;
00275             case 6:
00276                 if (wybrany_index + 1 == ilosc) wybrany_index--;
00277                 if (ilosc > 0) usun_z_poczatku();
00278                 break;
00279             case 7:
00280                 if (ilosc > 0) usun_z_konca();
00281                 break;
00282             case 8:
00283                 cout << "Podaj index: ";
00284                 cin >> wartosc2;
00285 
00286                 usun_z_indexu(wartosc2);
00287                 break;
00288             case 9:
00289                 widok = 1;
00290                 break;
00291             case 10:
00292                 widok = 2;
00293                 break;
00294             case 11:
00295                 widok = 0;
00296                 break;
00297             }
```

```
00298
00299         } while (komenda != 0 && ilosc != 0);
00300     }
```

4.5 E:/ProgramowanieZaawansowane/ZuchowiczDominik.ListaDwukierunkowa/↩ Example/Example/ListaDwukierunkowa.h File Reference

```
#include <string>
```

Classes

- struct Wezel

  Struktura reprezentujaca pojedynczy wezel listy dwukierunkowej.
- class ListaDwukierunkowa

  Klasa reprezentujaca liste dwukierunkowa.

4.6 ListaDwukierunkowa.h

Go to the documentation of this file.

```
00001 #pragma once
00002 using namespace std;
00003 #include <string>
00004
00010 struct Wezel {
00011     int dane;
00012     Wezel* nastepny;
00013     Wezel* poprzedni;
00018     Wezel() : dane(NULL), poprzedni(nullptr), nastepny(nullptr) {}
00019 };
00020
00026 class ListaDwukierunkowa {
00027 private:
00028     Wezel* poczatek;
00029     Wezel* koniec;
00030     int ilosc;
00032 public:
00036     ListaDwukierunkowa() : poczatek(nullptr), koniec(nullptr), ilosc(0) {}
00037
00041     ~ListaDwukierunkowa() {}
00042
00046     void wyswietl_od_poczatku();
00047
00051     void wyswietl_od_konca();
00052
00059     Wezel* dodaj_na_poczatek(int wartosc);
00060
00067     Wezel* dodaj_na_koniec(int wartosc);
00068
00076     Wezel* dodaj_na_index(int wartosc, int index);
00077
00081     void usun_z_poczatku();
00082
00086     void usun_z_konca();
00087
00093     void usun_z_indexu(int index);
00094
00101     string wezel(int index);
00102
00108     void Widok();
00109 };
```

Index