Algorytm listy dwukierunkowej

Generated by Doxygen 1.15.0

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 documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 ListaDwuKierunkowa Class Reference

Klasa listy dwukierunkowej.

`#include <lista-dwu-kierunkowa.h>`

Public Member Functions

- **ListaDwuKierunkowa** ()

  Konstruktor inicjalizujący pustą listę dwukierunkową. Ustawia wskaźniki przod i tyl na nullptr, co oznacza brak elementów początkowych.
- void **wyswietl** ()

  Wyświetla elementy listy od początku do końca. Iteruje po kolejnych węzłach i wypisuje przechowywane wartości. Jeżeli lista jest pusta, informuje o braku elementów.
- void **wyswietlNaOdwrot** ()

  Wyświetla elementy listy od końca do początku. Iteruje od węzła tyl do przodu, wypisując wartości w odwrotnej kolejności. Jeżeli lista jest pusta, informuje o braku elementów.
- void dodajNaPrzod (int x)

  Dodaje nowy element na początek listy. Tworzy węzeł Powiazanie i dołącza go przed dotychczasowym przodem listy. Jeżeli lista była pusta, nowy element staje się jednocześnie przodem i tyłem.
- void dodajNaTyl (int x)

  Dodaje nowy element na koniec listy. Tworzy węzeł Powiazanie i dołącza go za dotychczasowym tyłem listy. Jeżeli lista była pusta, nowy element pełni rolę przodu oraz tyłu.
- void **usunPrzod** ()

  Usuwa element na początku listy. Zwalnia pierwszy węzeł i aktualizuje wskaźnik przod. Gdy po operacji lista jest pusta, zeruje również wskaźnik tyl.
- void **usunTyl** ()

  Usuwa element na końcu listy. Zwalnia ostatni węzeł i aktualizuje wskaźnik tyl. Jeżeli po usunięciu lista jest pusta, zeruje także wskaźnik przod.
- void dodajPodIndeks (int val, int index)

  Dodaje nowy element pod wskazanym indeksem listy. Dla indeksu równego zero wstawia węzeł na początek, a gdy pozycja wykracza poza listę dołącza element na końcu.
- void usunIndeks (int index)

  Usuwa element znajdujący się pod danym indeksem. Dla indeksu równego zero usuwa pierwszy element; brak wskazanego węzła sygnalizuje komunikatem o błędzie.
- void wyswietlNastepny (int index)

Wyświetla element następujący po węźle o podanym indeksie. Gdy istnieje kolejny element, wypisuje jego wartość, w przeciwnym razie informuje o braku następnika.

- void wyswietlPoprzedni (int index)

  Wyświetla element poprzedzający węzeł o podanym indeksie. Gdy istnieje poprzednik, wypisuje jego wartość, a w przeciwnym wypadku informuje o jej braku.
- void **wyczysc** ()

  Usuwa wszystkie elementy z listy. Iteracyjnie zwalnia węzły rozpoczynając od przodu, aż struktura będzie pusta.
- void **test** ()

  Testowo tworzy dwuelementowy układ połączeń. Dodaje dwa węzły, łączy je ze sobą i przypisuje jako przód oraz tył listy.

3.1.1 Detailed Description

Klasa listy dwukierunkowej.

3.1.2 Member Function Documentation

3.1.2.1 dodajNaPrzod()

```
void ListaDwuKierunkowa::dodajNaPrzod (
            int x)
```

Dodaje nowy element na początek listy. Tworzy węzeł Powiazanie i dołącza go przed dotychczasowym przodem listy. Jeżeli lista była pusta, nowy element staje się jednocześnie przodem i tyłem.

Parameters

x	Wartość przypisana do nowego elementu.

3.1.2.2 dodajNaTyl()

```
void ListaDwuKierunkowa::dodajNaTyl (
            int x)
```

Dodaje nowy element na koniec listy. Tworzy węzeł Powiazanie i dołącza go za dotychczasowym tyłem listy. Jeżeli lista była pusta, nowy element pełni rolę przodu oraz tyłu.

Parameters

x	Wartość przypisana do nowego elementu.

3.1.2.3 dodajPodIndeks()

```
void ListaDwuKierunkowa::dodajPodIndeks (
            int val,
            int index)
```

Dodaje nowy element pod wskazanym indeksem listy. Dla indeksu równego zero wstawia węzeł na początek, a gdy pozycja wykracza poza listę dołącza element na końcu.

Parameters

val	Wartość przypisana do nowego elementu.
index	Indeks, pod którym należy umieścić nowy węzeł.

3.1.2.4 usunIndeks()

```
void ListaDwuKierunkowa::usunIndeks (
            int index)
```

Usuwa element znajdujący się pod danym indeksem. Dla indeksu równego zero usuwa pierwszy element; brak wskazanego węzła sygnalizuje komunikatem o błędzie.

Parameters

index	Indeks elementu przeznaczonego do usunięcia.

3.1.2.5 wyswietlNastepny()

```
void ListaDwuKierunkowa::wyswietlNastepny (
            int index)
```

Wyświetla element następujący po węźle o podanym indeksie. Gdy istnieje kolejny element, wypisuje jego wartość, w przeciwnym razie informuje o braku następnika.

Parameters

index	Indeks elementu, dla którego sprawdzany jest następnik.

3.1.2.6 wyswietlPoprzedni()

```
void ListaDwuKierunkowa::wyswietlPoprzedni (
            int index)
```

Wyświetla element poprzedzający węzeł o podanym indeksie. Gdy istnieje poprzednik, wypisuje jego wartość, a w przeciwnym wypadku informuje o jej braku.

Parameters

index	Indeks elementu, dla którego sprawdzany jest poprzednik.

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

- lista-dwu-kierunkowa.h
- lista-dwu-kierunkowa.cpp

3.2 Powiazanie Struct Reference

Struktura reprezentująca powiązanie w liście dwukierunkowej.

Public Member Functions

- **Powiazanie** (int wartosc)

Public Attributes

- int **dane**
- Powiazanie * **nastepny**
- Powiazanie * **poprzedni**

3.2.1 Detailed Description

Struktura reprezentująca powiązanie w liście dwukierunkowej.

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

- powiazanie.cpp

Chapter 4

File Documentation

4.1 lista-dwu-kierunkowa.h

```
00001 #pragma once
00002 #include "powiazanie.cpp"
00003
00007
00008 class ListaDwuKierunkowa {
00009 private:
00010     Powiazanie* przod;
00011     Powiazanie* tyl;
00012
00013 public:
00014     ListaDwuKierunkowa();
00015
00016     void wyswietl();
00017
00018     void wyswietlNaOdwrot();
00019
00020     void dodajNaPrzod(int x);
00021
00022     void dodajNaTyl(int x);
00023
00024     void usunPrzod();
00025
00026     void usunTyl();
00027
00028     void dodajPodIndeks(int val, int index);
00029
00030     void usunIndeks(int index);
00031
00032     void wyswietlNastepny(int index);
00033
00034     void wyswietlPoprzedni(int index);
00035
00036     void wyczysc();
00037
00038     void test();
00039 };
00040
```

4.2 powiazanie.cpp

```
00001 #pragma once
00002
00006
00007 struct Powiazanie {
00008     int dane;
00009     Powiazanie* nastepny;
00010     Powiazanie* poprzedni;
00011
00012     Powiazanie(int wartosc) : dane(wartosc), nastepny(nullptr), poprzedni(nullptr) {}
00013 };
```

Index