Lista dwukierunkowa w C++

Generated by Doxygen 1.12.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

Data Structure Documentation

2.1 DubleLinkedList Class Reference

Klasa reprezentująca listę dwukierunkową.

Public Member Functions

- DubleLinkedList ()
  Konstruktor klasy, inicjalizujący pustą listę.
- void addRandomToHead ()
  Dodaje losowy element na początek listy.
- void addRandomToBack ()
  Dodaje losowy element na koniec listy.
- void insertRandomAt (int index)
  Wstawia losowy element pod wskazany indeks.
- void removeFromHead ()
  Usuwa element z początku listy.
- void removeFromBack ()
  Usuwa element z końca listy.
- void removeAt (int index)
  Usuwa element pod danym indeksem.
- void display ()
  Wyświetla całą listę.
- void displayReverse ()
  Wyświetla listę w odwrotnej kolejności.
- void clear ()
  Czyści całą listę.
- ~DubleLinkedList ()
  Destruktor, który automatycznie wywołuje funkcję czyszczącą.

2.1.1 Detailed Description

Klasa reprezentująca listę dwukierunkową.

Klasa obsługuje operacje dodawania, usuwania, wyświetlania i czyszczenia elementów listy.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 DubleLinkedList()

```
DubleLinkedList.DubleLinkedList () [inline]
```

Konstruktor klasy, inicjalizujący pustą listę.

Ustawia head i back na nullptr, ponieważ lista początkowo jest pusta. Inicjalizuje również generator liczb losowych. < Ustawia ziarno dla generatora liczb losowych

2.1.2.2 ∼DubleLinkedList()

```
DubleLinkedList.∼DubleLinkedList () [inline]
```

Destruktor, który automatycznie wywołuje funkcję czyszczącą.

< Czyści listę przy wywołaniu destruktora

2.1.3 Member Function Documentation

2.1.3.1 addRandomToBack()

```
void DubleLinkedList.addRandomToBack () [inline]
```

Dodaje losowy element na koniec listy.

Generuje losową wartość i tworzy nowy węzeł, który jest dodawany na koniec listy. < Losuje wartość w zakresie 0-99

< Tworzy nowy węzeł o wygenerowanej wartości

< Ustawia head i back na newNode, jeśli lista jest pusta

< Nowy węzeł staje się nowym ostatnim elementem

< Wskaźnik next starego ostatniego elementu wskazuje na nowy węzeł

< Ustawia nowy węzeł jako back

2.1.3.2 addRandomToHead()

```
void DubleLinkedList.addRandomToHead () [inline]
```

Dodaje losowy element na początek listy.

Generuje losową wartość i tworzy nowy węzeł, który jest dodawany na początek listy. < Losuje wartość w zakresie 0-99

< Tworzy nowy węzeł o wygenerowanej wartości

< Ustawia head i back na newNode, jeśli lista jest pusta

< Nowy węzeł staje się nową głową

< Wskaźnik prev starej głowy wskazuje na nowy węzeł

< Ustawia nowy węzeł jako head

2.1.3.3 clear()

```
void DubleLinkedList.clear ()  [inline]
```

Czyści całą listę.

Usuwa wszystkie elementy z listy, używając metody removeFromHead(). < Usuwa każdy element zaczynając od początku

2.1.3.4 display()

```
void DubleLinkedList.display ()  [inline]
```

Wyświetla całą listę.

Przechodzi przez listę i wyświetla dane dla każdego elementu. < Przechodzi przez listę od head do back

2.1.3.5 displayReverse()

```
void DubleLinkedList.displayReverse ()  [inline]
```

Wyświetla listę w odwrotnej kolejności.

Przechodzi przez listę od back do head i wyświetla dane dla każdego elementu. < Wyświetla dane w odwrotnej kolejności

2.1.3.6 insertRandomAt()

```
void DubleLinkedList.insertRandomAt (
            int index)  [inline]
```

Wstawia losowy element pod wskazany indeks.

Generuje losową wartość i wstawia nowy węzeł w odpowiednim miejscu w liście.

Parameters

index	Indeks, pod którym ma być wstawiony nowy element

< Losuje wartość w zakresie 0-99

< Dodaje element na początek listy

< Szuka węzła na pozycji index - 1

< Indeks jest poza zakresem

< Ustawia wskaźniki nowego węzła

< Ustawia wskaźnik prev następnego węzła

< Ustawia nowy węzeł jako back, jeśli jest ostatni

< Ustawia wskaźnik next dla węzła na nowy węzeł

2.1.3.7 removeAt()

```
void DubleLinkedList.removeAt (
            int index)  [inline]
```

Usuwa element pod danym indeksem.

Sprawdza, czy indeks jest poprawny, a następnie usuwa odpowiedni element.

Parameters

index	Indeks elementu do usunięcia

< Usuwa element z początku listy

< Szuka węzła na podanym indeksie

< Indeks jest poza zakresem

< Dostosowuje wskaźniki sąsiednich węzłów

< Aktualizuje wskaźnik back, jeśli usuwany element to back

< Usuwa węzeł

2.1.3.8 removeFromBack()

```
void DubleLinkedList.removeFromBack ()  [inline]
```

Usuwa element z końca listy.

Sprawdza, czy lista nie jest pusta, a następnie usuwa ostatni element. < Sprawdza, czy lista nie jest pusta

< Ustawia back na poprzedni węzeł

< Ustawia wskaźnik next na nullptr

< Ustawia head na nullptr, jeśli lista jest pusta

< Usuwa stary węzeł

2.1.3.9 removeFromHead()

```
void DubleLinkedList.removeFromHead ()  [inline]
```

Usuwa element z początku listy.

Sprawdza, czy lista nie jest pusta, a następnie usuwa pierwszy element. < Sprawdza, czy lista nie jest pusta

< Ustawia head na następny węzeł

< Ustawia wskaźnik prev na nullptr

< Ustawia back na nullptr, jeśli lista jest pusta

< Usuwa stary węzeł

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

- Class1.cs

2.2 Node Struct Reference

Struktura reprezentująca węzeł w liście dwukierunkowej.

Public Member Functions

- Node (int value)
  Konstruktor węzła.

Data Fields

- int **data**
  Wartość węzła.
- Node ∗ **prev**
  Wskaźnik na poprzedni węzeł
- Node ∗ **next**
  Wskaźnik na następny węzeł

2.2.1 Detailed Description

Struktura reprezentująca węzeł w liście dwukierunkowej.

Węzeł zawiera dane oraz wskaźniki na poprzedni i następny węzeł.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 Node()

```
Node.Node (
            int value)  [inline]
```

Konstruktor węzła.

Parameters

value	Wartość, która będzie przechowywana w węźle

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

- Class1.cs