МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Алгоритми та структури даних»**

**на тему: «Динамічні лінійними структури даних»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Куроп’ятник О. С.

Дніпро, 2020

**Лабораторна робота №3**

**Тема.** Динамічні лінійними структури даних.

**Постановка задачі**

1.Індивідуальна постановка:

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант | Структура | Операції | Інформаційна частина |
| 7 | Однозвязний кільцевий впорядкований список | Додавання, видалення за ключем з поверненням елементу, пошук за позицією від заданої, перегляд елементів | Слово зі словника |

**Текст програми**

Word.java

package com.company;  
  
public class Word {  
 public String data;  
 public String idata;  
  
 public Word(String word, String word\_RU) {  
 this.data = word;  
 this.idata = word\_RU;  
 }  
  
 @Override  
 public String toString() {  
 return data + " - " + idata;  
 }  
}

Link.java

package com.company;  
  
  
class LinkList {  
 private static class Link {  
 Word data;  
 Link next; // посилання на наступний елемент списку  
  
 public Link(String word, String word\_RU) {  
 this.data = new Word(word, word\_RU);  
 }  
  
 public void displayLink() { System.*out*.println(" " + data); }  
 public void displayFind() { System.*out*.println("Ми знайшли слово: " + data); }  
 public void displayDelete() { System.*out*.println("Ми видалили: " + data); }  
 }  
  
 Link first;  
  
 public LinkList() {  
 first = null;  
 }  
  
 public void insertFirst(String word, String word\_RU) {  
 Link newLink = new Link(word, word\_RU); //створюємо новий елемент списку  
 newLink.next = first; //вказали на старий "перший" елемент і таким чином зробили його другим  
 first = newLink; // позначили створений елемент, як перший  
 }  
  
 public Link find(String key) { // пошук елемента с заданим ключем  
 Link current = first; // починаємо пошук з першого елемента  
 Link temp = null;  
  
 while (current != null) {  
 if(current.data.data.equals(key)){  
 current.displayFind();  
 temp = current;  
 break;  
 }  
  
 current = current.next;  
 }  
 return temp;  
 }  
  
 public Link delete(String key) { //видалення елементу по заданому ключу  
 Link current = first; // Текущий  
 Link previous = first; // предыдущий  
  
  
 while (current != null) { // пошук елемента  
  
 if(current.data.data.equals(key)){  
 if (current == first) { // якщо шуканий елемент - перший  
 first = first.next;  
 } else {  
 previous.next = current.next; // якщо шуканий елемент лежить всередині списку, обійдемо його  
 }  
 current.displayDelete();  
 }  
  
 if (current.next == null) {  
 return null; // елемент не знайдено  
 } else {  
 previous = current; // перейти до наступного елементу  
 current = current.next;  
 }  
 }  
 return current;  
 }  
  
 public void displayList() {  
 Link current = first;  
 System.*out*.println("List (first --> last): ");  
 while (current != null) { //доки список не закінчився, відображаємо дані  
 current.displayLink(); // відображаємо дані елементу, на якому знаходимося  
 current = current.next; // переходимо до наступного елементу  
 }  
 }  
}

Main.java

package com.company;  
  
import java.util.Scanner;  
/\*Однозв’язний кільцевий впорядкований список:  
 -додавання  
 -видалення за ключем з поверненням елементу  
 -пошук за позицією від заданої  
 -перегляд елементів\*/  
public class Main {  
  
 public static void main(String[] args) {  
 // write your code here  
 LinkList List = new LinkList();  
 List.insertFirst("hi", "привет");  
 List.insertFirst("drift", "дрифт");  
 List.insertFirst("problem", "проблема");  
 List.insertFirst("task", "задача");  
 int menu\_ = 0;  
 do {  
 Scanner menu = new Scanner(System.*in*);  
 System.*out*.println("1.Додавання");  
 System.*out*.println("2.Видалення за ключем");  
 System.*out*.println("3.Пошук за позицією");  
 System.*out*.println("4.Перегляд");  
 System.*out*.println("5.Вихід");  
  
 menu\_ = menu.nextInt();  
 switch (menu\_) {  
 case 1: {  
 String name, name2;  
 System.*out*.println("Введіть слово: ");  
 Scanner scannerStringName = new Scanner(System.*in*);  
 name = scannerStringName.nextLine();  
 System.*out*.println("Введіть переклад слово: ");  
 Scanner scannerStringName2 = new Scanner(System.*in*);  
 name2 = scannerStringName2.nextLine();  
 List.insertFirst(name, name2);  
 break;  
 }  
 case 2: {  
 String key\_1;  
 System.*out*.println("Введіть слово: ");  
 Scanner scannerString\_1 = new Scanner(System.*in*);  
 key\_1 = scannerString\_1.nextLine();  
 List.delete(key\_1);  
 break;  
 }  
 case 3: {  
 String key\_2;  
 System.*out*.println("Введіть слово: ");  
 Scanner scannerString\_2 = new Scanner(System.*in*);  
 key\_2 = scannerString\_2.nextLine();  
 List.find(key\_2);  
 break;  
 }  
 case 4: {  
 List.displayList();  
 break;  
 }  
 }  
 }while (menu\_ != 5);  
 }  
}

Висновок: у цій лабораторній роботі я з’ясував недоліки та переваги однозв’язного списку.

Переваги:

На відміну від масиву є можливість додавання вузол у кінець списку, видаляти вузол та вставляти вузол в середину. У масиві часто треба буде створювати масив більшого або меншого розміру (якщо немає вільного місця) перезаписувати старий масив у новий і тільки потім можна додати або видалити елемент.

Недоліки:

У списку немає по індексованого доступу до елементів, через що ми витрачаємо зайвий час на перехід по списку ( пошук, видалення, додавання). Також ми використовуємо більше об’єму пам’яті за рахунок покажчиків на наступний вузол.