**Державний вищий навчальний заклад**

**Ужгородський національний університет**

**Факультет інформаційних технологій**

**Практична робота №2**

**Односпрямовані та двоспрямовані списки**

Виконав студент 1 курсу

спеціальності “Інженерія

програмного забезпечення”

Гарабаджіу Крістіан

**Ужгород-2025**

**Мета:** набути практичних навичок створення та застосування односпрямованих та двоспрямованих списків.

Завдання до роботи:

1. Написати програму, у якій потрібно створити лінійний односпрямований список та виконати дії згідно з власним варіантом завдання табл. 1.1.
2. Написати програму, у якій потрібно створити двоспрямований список та виконати дії згідно з власним варіантом завдання табл 1.2.
3. Оформити звіт про виконання роботи та завантажити його в системі електронного навчання ДВНЗ «УжНУ» в установлений термін.
4. Підготувати відповіді на питання до лекції з односпрямованого та двоспрямованого списку.

**Хід роботи:**

1. Створити лінійний односпрямований список із дійсних чисел. Визначити середнє значення елементів списку, які мають значення менше або дорівнює 15. Видалити зі списку елементи, які більше 25

class Element {

constructor(number) {

this.data = number;

this.next = null;

}  
}

class specList {

constructor () {

this.head = null;

this.size = 0;

}

add (el, pos){

if(this.head === null) {

this.head = el;

this.size++;

}else if(pos === undefined){

var currentEl = this.head;

while (currentEl.next !== null) {

currentEl = currentEl.next;

}

currentEl.next = el;

this.size++;

}else if(pos < this.size && pos > 0){

var currentEl = this.head;

var mypos = 0;

while (mypos < pos - 1){

mypos++;

currentEl = currentEl.next;

}

el.next = currentEl.next;

currentEl.next = el;

this.size++;

}else{

confirm.log("Вказано неправильний індекс. Елемент не додано до списку");

}

}

delete (pos){

if(pos === undefined && this.size !=0){

var currentEl = this.head;

var prevCurrentEl;

while (currentEl.next !== null) {

prevCurrentEl = currentEl;

currentEl = currentEl.next;

}

prevCurrentEl.next = null;

this.size--;

}else if(pos < this.size && pos > 0 && this.size != 0){

if(pos === 0){

this.head = this.head.next;

}else{

var currentEl = this.head;

var prevCurrentEl;

var mypos = 0;

while (mypos < pos){

mypos++;

prevCurrentEl = currentEl;

currentEl = currentEl.next;

}

prevCurrentEl.next = currentEl.next;

}

this.size--;

}else if (this.size === 0){

console.log("Неможливо видалити елемент з порожнього списку. Елемент не видалено");

}else{

console.log("Вказано неправильний індекс. Елемент не видалено");

}

}

midM15 (withCheck){

var currentEl = this.head;

var sumData = [null, 0];

while(currentEl){

if(currentEl.data <= 15){

sumData[0] += currentEl.data;

sumData[1]++;

if(withCheck){console.log(currentEl.data);}

}

currentEl = currentEl.next;

}

if(sumData[0] === null){

console.log("В списку немає елементів, які б були <= 15");

}else{

console.log(`Середнє значення елментів <= 15: ${sumData[0]/sumData[1]}`);

}

}

delB25 (withCheck){

var currentEl = this.head;

while (currentEl && currentEl.next !== null) {

if(currentEl.data > 25 && currentEl.data === this.head.data && currentEl.next === this.head.next){

this.head = this.head.next;

this.size--;

}else{

if(currentEl.next.data > 25){

if(withCheck){console.log(currentEl.next.data);}

currentEl.next = currentEl.next.next;

this.size--;

}else{

currentEl = currentEl.next;

}

}

}

console.log("Елементи, які більше 25, видалено!")

}

viewList (){

var res = "";

var currentEl = this.head;

while (currentEl) {

res += currentEl.data + " ";

currentEl = currentEl.next;

}

console.log(res.trim());

}

clear (){

this.head = null;

this.size = 0;

}  
}

var mySpecList = new specList();const randomNumbers = [12, 45, 8, 33, 27, 50, 3, 19, 41, 23, 9, 48, 14, 37, 6, 29, 2, 44, 30, 11, 21, 39, 4, 16, 35, 47, 25, 10, 42, 7];

randomNumbers.forEach(el => mySpecList.add(new Element(el)));

mySpecList.viewList();

mySpecList.add(new Element(88), 2);

mySpecList.viewList();

mySpecList.delete();

mySpecList.viewList();

mySpecList.delete(2);

mySpecList.viewList();

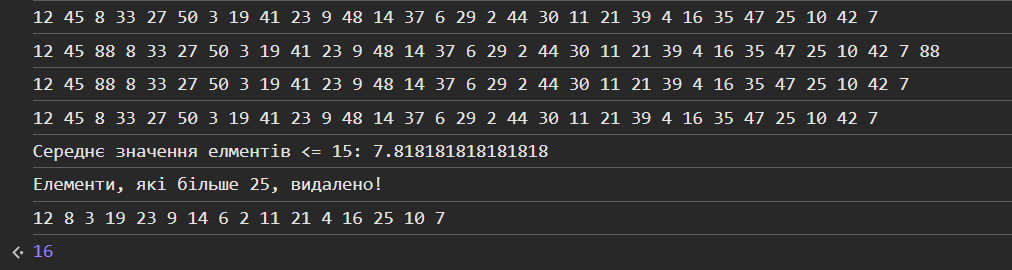
mySpecList.midM15();

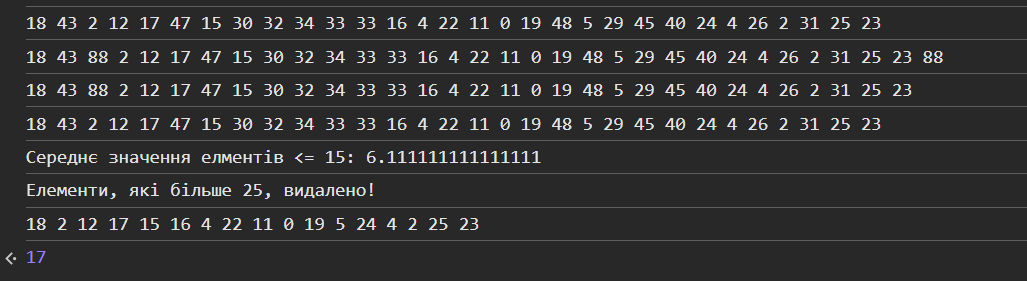
mySpecList.delB25();

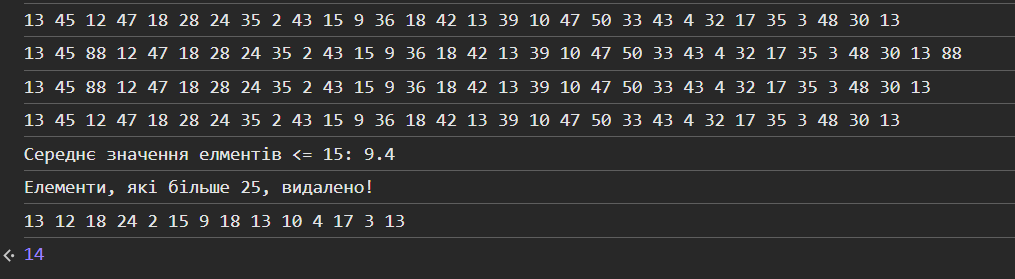
mySpecList.viewList();

mySpecList.size

//[18, 43, 2, 12, 17, 47, 15, 30, 32, 34, 33, 33, 16, 4, 22, 11, 0, 19, 48, 5, 29, 45, 40, 24, 4, 26, 2, 31, 25, 23]  
//[13, 45, 12, 47, 18, 28, 24, 35, 2, 43, 15, 9, 36, 18, 42, 13, 39, 10, 47, 50, 33, 43, 4, 32, 17, 35, 3, 48, 30, 13]







1. Дано покажчик P1 на перший елемент двозв’язного списку, який має не менш двох елементів. Видалити зі списку усі елементи з непарними номерами та вивести покажчик на перший елемент перетвореного списку.

class Element {

constructor(number) {

this.data = number;

this.next = null;

this.prev = null;

}  
}

class specList {

constructor() {

this.head = null;

this.size = 0;

}

add(el, pos){

if(this.head === null) {

this.head = el;

this.size++;

}else if(pos === undefined){

var currentEl = this.head;

while (currentEl.next !== null) {

currentEl = currentEl.next;

}

currentEl.next = el;

currentEl.next.prev = currentEl;

this.size++;

}else if(pos < this.size && pos > 0){

var currentEl = this.head;

var mypos = 0;

while (mypos < pos - 1){

mypos++;

currentEl = currentEl.next;

}

el.next = currentEl.next;

currentEl.next = el;

currentEl.next.prev = currentEl;

currentEl.next.next.prev = currentEl.next

this.size++;

}else{

confirm.log("Вказано неправильний індекс. Елемент не додано до списку");

}

}

delete(pos){

if(pos === undefined && this.size !=0){

var currentEl = this.head;

while (currentEl.next !== null) {

currentEl = currentEl.next;

}

currentEl.prev.next = null;

this.size--;

}else if(pos < this.size && pos > 0 && this.size != 0){

if(pos === 0){

this.head = this.head.next;

}else{

var currentEl = this.head;

var mypos = 0;

while (mypos < pos){

mypos++;

currentEl = currentEl.next;

}

currentEl.prev.next = currentEl.next;

}

this.size--;

}else if (this.size === 0){

console.log("Неможливо видалити елемент з порожнього списку. Елемент не видалено");

}else{

console.log("Вказано неправильний індекс. Елемент не видалено");

}

}

Delesec(reg = 1){

if(reg == 2){this.head = this.head.next; this.size--;}

var currentEl = this.head;

while (currentEl) {

if(currentEl.next != null){

currentEl.next = currentEl.next.next;

currentEl = currentEl.next;  
 this.size--;

}else{

currentEl = currentEl.next;

}

}

}

viewList(){

var res = "";

var currentEl = this.head;

while (currentEl) {

res += currentEl.data + " ";

currentEl = currentEl.next;

}

console.log(res.trim());

}

clear(){

this.head = null;

this.size = 0;

}

}

var mySpecList = new specList();const randomNumbers = [12, 45, 8, 33, 27, 50, 3, 19, 41, 23, 9, 48, 14, 37, 6, 29, 2, 44, 30, 11, 21, 39, 4, 16, 35, 47, 25, 10, 42, 7];

randomNumbers.forEach(el => mySpecList.add(new Element(el)));

mySpecList.viewList();

mySpecList.add(new Element(88), 2);

mySpecList.add(new Element(88));

mySpecList.viewList();

mySpecList.delete();

mySpecList.viewList();

mySpecList.delete(2);

mySpecList.viewList();

mySpecList.Delesec(2);

mySpecList.viewList();

console.log(mySpecList.size);

mySpecList.clear();

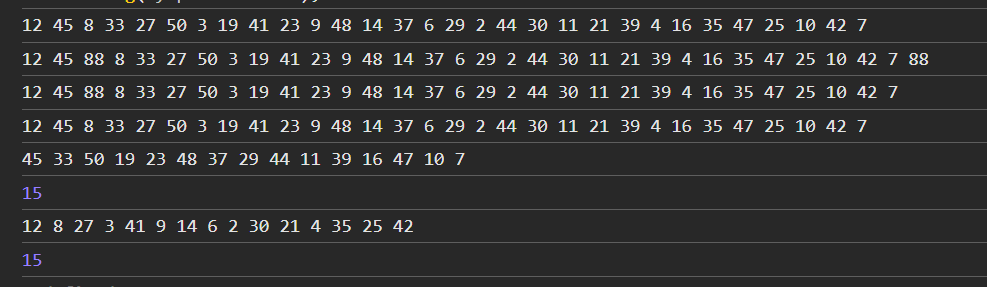
randomNumbers.forEach(el => mySpecList.add(new Element(el)));

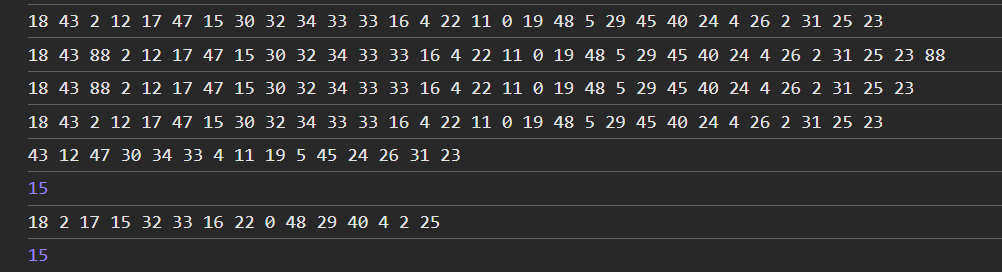
mySpecList.Delesec();

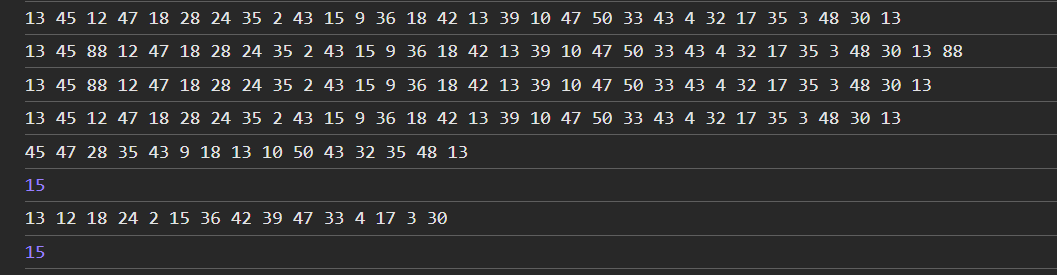
mySpecList.viewList();

console.log(mySpecList.size);

//[18, 43, 2, 12, 17, 47, 15, 30, 32, 34, 33, 33, 16, 4, 22, 11, 0, 19, 48, 5, 29, 45, 40, 24, 4, 26, 2, 31, 25, 23]  
//[13, 45, 12, 47, 18, 28, 24, 35, 2, 43, 15, 9, 36, 18, 42, 13, 39, 10, 47, 50, 33, 43, 4, 32, 17, 35, 3, 48, 30, 13]







**Висновок:**

Виконана робота продемонструвала основні принципи створення та роботи з односпрямованими і двоспрямованими зв’язаними списками, які є важливими структурами даних у програмуванні. Методи додавання (add) та видалення (delete) є “перевантаженими”, бо адаптовані також для вставки елементів за індексом. Обчислення середнього значення для елементів списку, що задовольняють певні умови (midM15), продемонстрували важливість коректного оновлення покажчиків (особливо для однозв’язного списку), щоб не втрачати доступ до частини списку, а також важливість перевірки наступних елементів на null. Двоспрямовані списки, на відміну від односпрямованих, мають перевагу у швидкому доступі як до наступного, так і до попереднього елемента, що значно спрощує виконання багатьох операцій, особливо видалення та вставки в середину списку. Загалом, використання динамічних структур дозволяє ефективно керувати пам’яттю та реалізовувати складні алгоритми, що є важливим для розробки програмного забезпечення.