**Державний вищий навчальний заклад  
Ужгородський національний університет  
Факультет інформаційних технологій**

ПРАКТИЧНА РОБОТА № 2  
**Тема:** Односпрямовані та двоспрямовані списки

Виконав студент  
І курсу спеціальності «Інженерія програмного забезпечення»

Романюк Артем

**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.

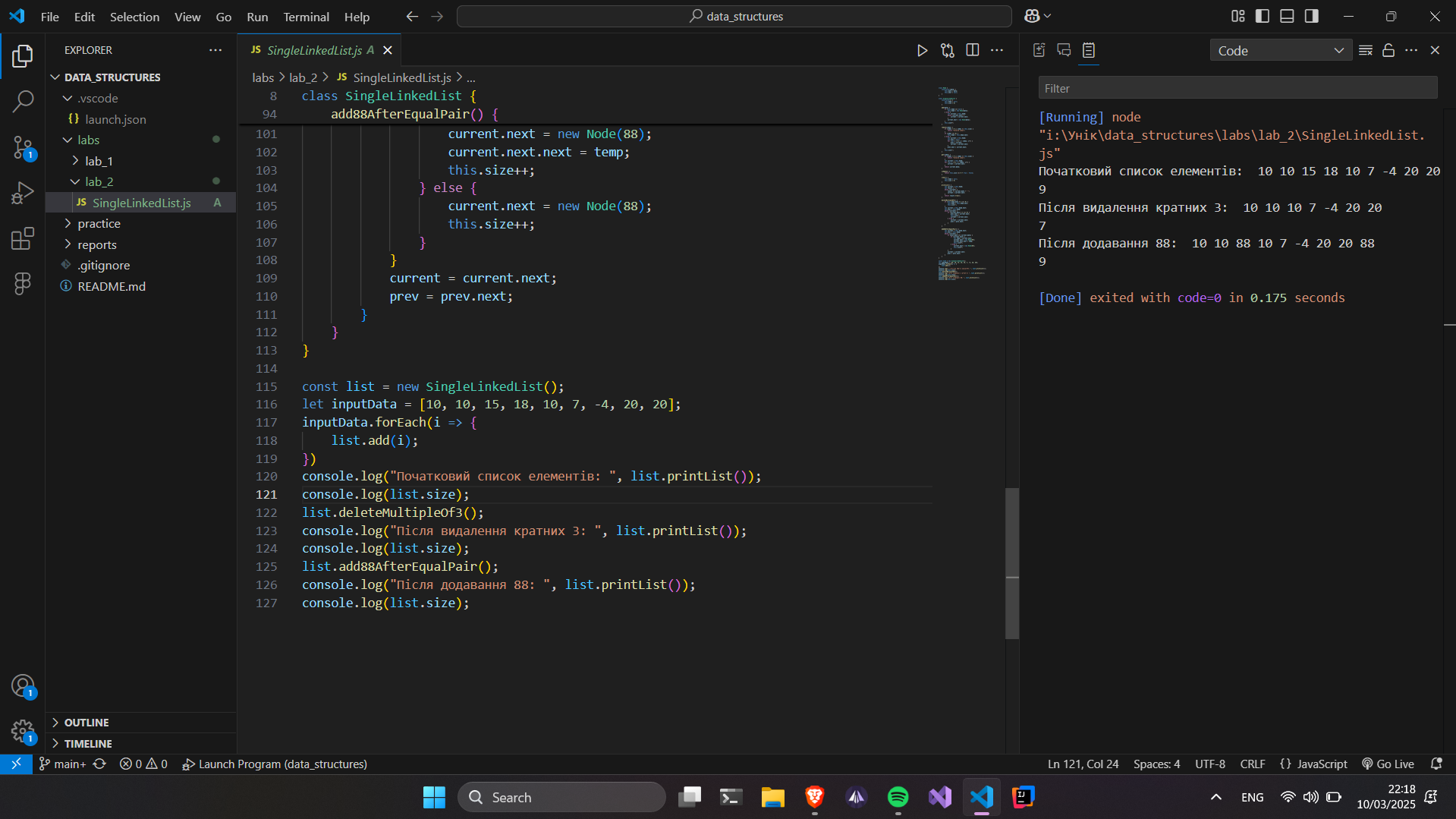
**Хід роботи:**

**Варіант 27**

1. Створити лінійний односпрямований список з цілих чисел. Видалити зі списку кожний елемент, який є кратний трьом. Додати до списку число 88 після кожної пари рівних чисел

[Реалізація в коді (github):](https://github.com/ArtemRomaniuk/data_structures/blob/main/labs/lab_2/SingleLinkedList.js)

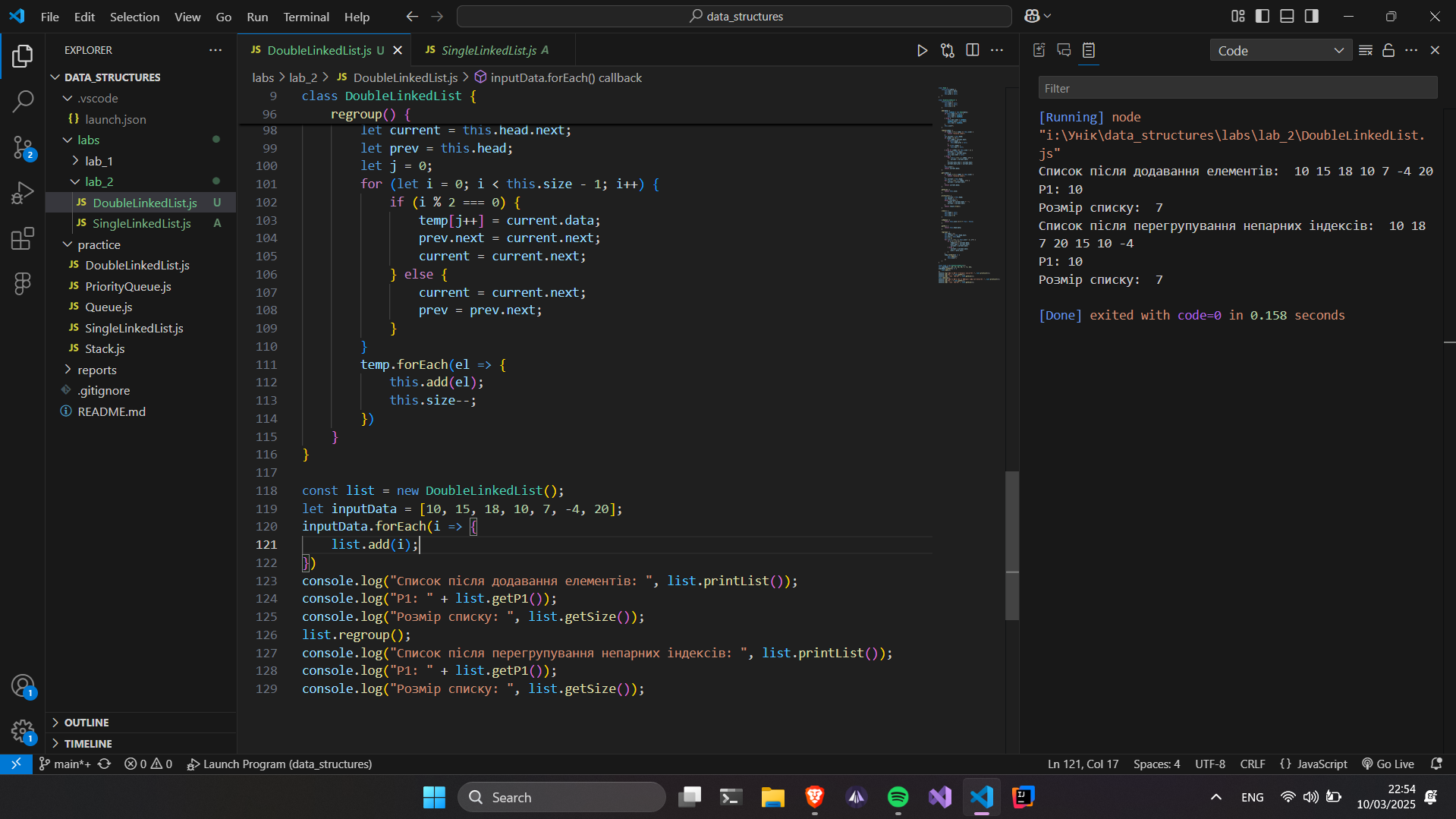
| class Node {  constructor(data) {  this.data = data;  this.next = null;  }  }  class SingleLinkedList {  constructor() {  this.head = null;  this.size = 0;  }  add(data) {  if (this.head === null) {  this.head = new Node(data);  } else {  let current = this.head;  while (current.next) {  current = current.next;  }  current.next = new Node(data);  }  this.size++;  }  remove(index) {  if (index < 0 || index >= this.size) {  return "Invalid index";  }  if (index === 0) {  this.head = this.head.next;  } else {  let current = this.head;  let prev = null;  for (let i = 0; i < index; i++) {  prev = current;  current = current.next;  }  prev.next = current.next;  }  this.size--;  }  get(index) {  if (index < 0 || index >= this.size) {  return "Invalid index";  }  let current = this.head;  for (let i = 0; i < index; i++) {  current = current.next;  }  return current.data;  }  isEmpty() {  return this.size === 0 ? true : false;  }  clear() {  this.head = null;  this.size = 0;  }  printList() {  let current = this.head;  let result = "";  while (current) {  result += current.data + " ";  current = current.next;  }  return result.trim();  }  deleteMultipleOf3() {  if (this.head.data % 3 === 0) {  this.head = this.head.next;  this.size--;  }  let current = this.head.next;  let prev = this.head;  while (current) {  if (current.data % 3 === 0) {  prev.next = current.next;  this.size--;  current = current.next;  } else {  current = current.next;  prev = prev.next;  }  }  }  add88AfterEqualPair() {  let current = this.head.next;  let prev = this.head;  while (current) {  if (prev.data === current.data) {  if (current.next) {  let temp = current.next;  current.next = new Node(88);  current.next.next = temp;  this.size++;  } else {  current.next = new Node(88);  this.size++;  }  }  current = current.next;  prev = prev.next;  }  }  }  const list = new SingleLinkedList();  let inputData = [10, 10, 15, 18, 10, 7, -4, 20, 20];  inputData.forEach(i => {  list.add(i);  })  console.log("Початковий список елементів: ", list.printList());  console.log(list.size);  list.deleteMultipleOf3();  console.log("Після видалення кратних 3: ", list.printList());  console.log(list.size);  list.add88AfterEqualPair();  console.log("Після додавання 88: ", list.printList());  console.log(list.size); |
| --- |



1. Дано покажчик P1 на перший елемент непорожнього двозв’язного списку. Перегрупувати його елементи, перемістити всі елементи з непарними номерами в кінець списку (в тому самому порядку) та вивести покажчик на перший елемент перетвореного списку.

[Реалізація в коді (github):](https://github.com/ArtemRomaniuk/data_structures/blob/main/labs/lab_2/DoubleLinkedList.js)

| class Node {  constructor(data) {  this.data = data;  this.next = null;  this.prev = null;  }  }  class DoubleLinkedList {  constructor() {  this.head = null;  this.tail = null;  this.size = 0;  }  add(data) {  const newNode = new Node(data);  if (this.head === null) {  this.head = newNode;  this.tail = newNode;  } else {  this.tail.next = newNode;  newNode.prev = this.tail;  this.tail = newNode;  }  this.size++;  }  remove(index) {  if (index < 0 || index >= this.size) {  return "Invalid index";  }  let current = this.head;  if (index === 0) {  this.head = current.next;  if (this.head) {  this.head.prev = null;  }  if (!this.head) {  this.tail = null;  }  } else if (index === this.size - 1) {  current = this.tail;  this.tail = current.prev;  this.tail.next = null;  } else {  for (let i = 0; i < index; i++) {  current = current.next;  }  current.prev.next = current.next;  current.next.prev = current.prev;  }  this.size--;  return current.data;  }  get(index) {  if (index < 0 || index >= this.size) {  return "Invalid index";  }  let current = this.head;  for (let i = 0; i < index; i++) {  current = current.next;  }  return current.data;  }  getSize() {  return this.size;  }  printList() {  let current = this.head;  let result = '';  while (current) {  result += current.data + " ";  current = current.next;  }  return result.trim();  }  clear() {  this.head = null;  this.tail = null;  this.size = 0;  }  isEmpty() {  return this.size === 0 ? true : false;  }  getP1() {  return this.head.data;  }  regroup() {  let temp = [];  let current = this.head.next;  let prev = this.head;  let j = 0;  for (let i = 0; i < this.size - 1; i++) {  if (i % 2 === 0) {  temp[j++] = current.data;  prev.next = current.next;  current = current.next;  } else {  current = current.next;  prev = prev.next;  }  }  temp.forEach(el => {  this.add(el);  this.size--;  })  }  }  const list = new DoubleLinkedList();  let inputData = [10, 15, 18, 10, 7, -4, 20];  inputData.forEach(i => {  list.add(i);  })  console.log("Список після додавання елементів: ", list.printList());  console.log("P1: " + list.getP1());  console.log("Розмір списку: ", list.getSize());  list.regroup();  console.log("Список після перегрупування непарних індексів: ", list.printList());  console.log("P1: " + list.getP1());  console.log("Розмір списку: ", list.getSize()); |
| --- |



**Висновок:**

У ході виконання лабораторної роботи було досліджено односпрямовані та двоспрямовані списки, їх реалізацію, основні операції та відмінності між ними.

Односпрямовані списки представляють собою набір вузлів, кожен з яких містить значення та посилання на наступний елемент. Проте основним недоліком односпрямованого списку є складність виконання деяких операцій, таких як видалення вузла без прямого доступу до попереднього елемента.

Двоспрямовані списки, на відміну від односпрямованих, містять у кожному вузлі два покажчики: на наступний і попередній елементи. Це дає змогу здійснювати швидке переміщення у двох напрямках, що значно спрощує виконання деяких операцій, зокрема видалення та вставку елементів. Проте така структура вимагає додаткової пам’яті для зберігання додаткових покажчиків.

Практичне застосування обох типів списків залежить від вимог конкретної задачі. Односпрямовані списки є ефективними у випадках, коли операції здійснюються лише в одному напрямку, тоді як двоспрямовані списки краще підходять для ситуацій, де важливий швидкий доступ у двох напрямках.

В результаті виконання роботи було отримано практичні навички реалізації та використання односпрямованих і двоспрямованих списків, що є важливим для ефективного використання структур даних у програмуванні.