ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ  
з дисципліни «Основи програмування на мові Javascript»  
студентки групи ПС-21-1  
Лисової Єлизавети Сергіївни  
кафедра комп’ютерних технологій, ДНУ  
2024/2025

1. Постановка задачі

Завдання 1  
Програма виводить варіант перестановки у якості завдання для користувача, зображує A,B,C,D,E за допомогою певних елементів керування на веб-сторінці, надає користувачеві можливість здійснити відповідну перестановку і побачити результат.

Завдання 2  
Програма малює задану область на канві (canvas) веб-сторінки засобами мови JavaScript. Програма надає користувачеві можливість вводити координати точки з веб-сторінки, малює траєкторію точки на її шляху до влучення, виводить підказки користувачеві на веб-сторінку.

Завдання 3  
Програма вводить послідовність з текстового файлу, відображує послідовність на веб-сторінці, упорядковує послідовність, відображує результат на веб-сторінці і виводить результат у текстовий файл. Програма надає користувачеві можливість вибрати файли в діалозі: файл з якого буде прочитано вхідну послідовність і файл, у який буде записано результуючу послідовність. Робота з файлами здійснюється через інтерфейс веб-сторінки (на відміну від завдання попередньої лабораторної роботи).

2. Опис розв’язку

Завдання 1:

Рішення базується на використанні HTML, CSS і JavaScript для створення інтерактивного інтерфейсу, який дозволяє користувачеві вибрати дві кнопки та поміняти їх тексти місцями.

1. **HTML:**  
   У розмітці створено структуру сторінки, яка містить:
   * Заголовок (<h1>), що пояснює функціонал.
   * Інструкцію для користувача (<p>).
   * Рядок кнопок (<div>), кожна з яких має подію onclick, прив’язану до функції handleClick.
   * Елемент для відображення статусу операцій (<p> з id="status").
2. **CSS:**  
   Додано стилі для візуалізації:
   * Центрування тексту і елементів на сторінці.
   * Зручне розташування кнопок з відступами.
   * Задання розміру та зовнішнього вигляду кнопок для інтуїтивного використання.
3. **JavaScript:**  
   Основна логіка реалізована у функції handleClick, яка:
   * Використовує змінну firstButton для збереження посилання на першу натиснуту кнопку.
   * Після натискання другої кнопки виконує обмін текстових значень між двома кнопками за допомогою тимчасової змінної temp.
   * Виводить повідомлення про вибір першої кнопки або про успішне виконання перестановки у відповідний елемент status.
   * Скидає значення firstButton для початку нового циклу вибору.
4. **Алгоритм роботи:**
   * При натисканні на кнопку викликається функція handleClick.
   * Якщо це перша кнопка (змінна firstButton має значення null), функція запам'ятовує її і виводить повідомлення.
   * Якщо вже вибрано першу кнопку, то відбувається обмін текстами між двома кнопками, а firstButton скидається в null.
5. **Додатковий функціонал:**  
   Для зручності користувача додано повідомлення про вибір першої кнопки і успішну перестановку, щоб уникнути помилок і покращити зворотний зв’язок.

Цей підхід забезпечує простоту коду, мінімальне використання ресурсів і зрозумілість для користувачів.

Завдання 2:

1. **HTML**:
   * Створено основу для гри, яка містить:
     + Заголовок і опис завдання.
     + Canvas для відображення ігрового поля, на якому буде розташована точка та цільова область.
     + Кнопки для керування точкою (вліво, вправо, вгору, вниз).
     + Блок для відображення підказок щодо поточного місця знаходження точки.
2. **CSS**:
   * Стилі оформлення сторінки для візуалізації:
     + Центрування елементів.
     + Встановлення розміру та рамки для canvas.
     + Оформлення кнопок і підказок для зручності користувача.
3. **JavaScript**:
   * **Малювання цільової області**: на canvas малюються три геометричні фігури, які разом утворюють цільову зону, в яку потрібно потрапити. Це вертикальні та горизонтальні смуги з різними кольорами, що перекриваються.
     + verticalStrip1 — вертикальна червона смуга.
     + horizontalStrip — горизонтальна зелена смуга.
     + verticalStrip2 — вертикальна синя смуга.
   * **Переміщення точки**: точка, що малюється на canvas, може переміщатися за допомогою кнопок. Її координати змінюються при натисканні на відповідну кнопку:
     + ⬅️ Вліво — змінює координату x на -10.
     + ➡️ Вправо — змінює координату x на +10.
     + ⬆️ Вверх — змінює координату y на -10.
     + ⬇️ Вниз — змінює координату y на +10.
   * **Перевірка попадання в ціль**: за допомогою функції isPointInTargetArea перевіряється, чи знаходиться точка в межах цільової області, яка визначається перехрестям трьох фігур.
   * **Відстань до центру**: за допомогою функції calculateDistance обчислюється відстань точки до центру області, що дозволяє надавати підказки щодо того, чи наближається користувач до цілі.
   * **Підказки користувачу**: кожного разу після переміщення точки додається підказка в блок "Історія спроб". Якщо точка потрапила в ціль, підказка повідомить про успішне попадання, а якщо ні — виводиться повідомлення про те, чи користувач наближається або віддаляється від цілі (тематичні підказки "Тепліше" та "Холодніше").
   * **Оновлення canvas**: після кожної спроби (переміщення точки) поле оновлюється, і малюється нове положення точки разом з цільовою областю.
4. **Алгоритм роботи:**
   * Гра починається з випадкової початкової позиції точки.
   * Користувач переміщує точку за допомогою кнопок, і кожен рух обчислюється щодо цільової області.
   * Після кожного руху програма надає інформацію, чи стає точка ближчою чи далі від цілі, і оновлює графіку на canvas.
   * Якщо точка потрапила в ціль, виводиться відповідне повідомлення, і гра завершується.
5. **Особливості:**
   * Використано HTML5 <canvas> для динамічного малювання ігрового поля.
   * Задано обмеження на переміщення точки за допомогою кнопок, щоб не вийти за межі canvas.
   * Оновлення підказок дозволяє користувачеві зрозуміти, як наблизитися до цілі.

Цей підхід забезпечує інтерактивний і цікавий спосіб керування точкою, а також дозволяє надавати корисні підказки для досягнення мети.

Завдання 3:

1. **HTML:**
   * Створено елементи для завантаження та відображення послідовності:
     + Поле для завантаження файлу (<input type="file" id="inputFile" accept=".txt">), через яке користувач може вибрати файл.
     + Кнопка для завантаження послідовності з файлу.
     + Блок для відображення елементів послідовності (<div id="sequence"></div>).
     + Кнопки для сортування послідовності та збереження результату в файл.
     + Опис інструкцій для користувача.
2. **CSS:**
   * Стилізація для зручності відображення елементів послідовності:
     + Використано Flexbox для вирівнювання елементів.
     + Кожен елемент має клас .sequence-item, щоб задати стиль (відступи, фоновий колір та межі).
     + Кнопки для взаємодії з додатком оформлені для зручного користування.
3. **JavaScript:**
   * **Завантаження послідовності**:
     + Функція loadSequence() відповідає за завантаження текстового файлу, який містить послідовність. При завантаженні тексту з файлу, розділеного пробілами чи новими рядками, ці елементи записуються в масив sequence.
   * **Відображення послідовності**:
     + Функція displaySequence() оновлює HTML, відображаючи елементи послідовності у вигляді клікабельних блоків. Кожен блок має обробник події для кліку, що дозволяє змінювати порядок елементів.
   * **Перестановка елементів**:
     + Користувач може натискати на елементи, щоб поміняти їх місцями. Для цього використовується функція handleSwap(index), яка:
       - Перший клік виділяє елемент, змінивши його фон на жовтий.
       - Другий клік переставляє вибрані елементи місцями в масиві sequence, а потім викликається displaySequence() для оновлення відображення.
   * **Сортування послідовності**:
     + Функція sortSequence() сортує елементи масиву в алфавітному порядку за допомогою методу sort(). Після сортування викликається displaySequence() для відображення оновленої послідовності.
   * **Збереження результату в файл**:
     + Функція saveSequence() створює текстовий файл з відсортованою послідовністю. Створюється об'єкт Blob, який перетворюється на URL для завантаження. В результаті користувач може завантажити відсортований файл як sorted\_sequence.txt.
4. **Алгоритм роботи:**
   * Після завантаження файлу з послідовністю, користувач може:
     + Переставляти елементи між собою, натискаючи на них.
     + Сортувати послідовність за допомогою кнопки "Упорядкувати".
     + Зберегти оновлену або відсортовану послідовність у файл за допомогою кнопки "Зберегти у файл".
5. **Особливості:**
   * Завдяки використанню функції FileReader в JavaScript, послідовність можна завантажити з текстового файлу.
   * Користувач має можливість мануально змінювати порядок елементів за допомогою простого інтерфейсу (натискання на елементи).
   * Всі зміни в послідовності відразу відображаються на екрані.
   * Для збереження результату користувач може завантажити новий файл з відсортованими елементами.

Цей підхід дозволяє створити інтерактивний додаток для роботи з текстовими послідовностями, надаючи користувачу можливість не тільки сортувати, але й змінювати порядок елементів вручну.

3. Вихідний текст програми розв’язку задачі (основні фрагменти з коментарями)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Перестановка кнопок</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 50px;

}

.button-row {

margin: 10px 0;

}

button {

padding: 10px 20px;

margin: 5px;

font-size: 18px;

cursor: pointer;

}

</style>

</head>

<body>

<h1>Перестановка кнопок</h1>

<p>Натисніть дві кнопки, щоб поміняти їх місцями.</p>

<div class="button-row" id="buttonRow">

<button onclick="handleClick(this)">A</button>

<button onclick="handleClick(this)">B</button>

<button onclick="handleClick(this)">C</button>

<button onclick="handleClick(this)">D</button>

<button onclick="handleClick(this)">E</button>

</div>

<p id="status" style="color: green; font-weight: bold;"></p>

<script>

let firstButton = null;

function handleClick(button) {

const status = document.getElementById("status");

if (!firstButton) {

// Запам'ятати першу кнопку

firstButton = button;

status.textContent = `Вибрано першу кнопку: ${button.textContent}`;

} else {

// Виконати перестановку

const secondButton = button;

const temp = firstButton.textContent;

firstButton.textContent = secondButton.textContent;

secondButton.textContent = temp;

status.textContent = `Переставлено кнопки: ${firstButton.textContent} та ${secondButton.textContent}`;

firstButton = null; // Скинути вибір

}

}

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Влучити в область</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 20px;

}

canvas {

border: 2px solid black;

margin: 20px auto;

display: block;

}

.controls {

margin-top: 20px;

}

button {

padding: 10px 20px;

margin: 5px;

font-size: 16px;

cursor: pointer;

}

#hints {

margin-top: 20px;

border: 1px solid #ccc;

padding: 10px;

max-width: 400px;

margin: 20px auto;

text-align: left;

}

</style>

</head>

<body>

<h1>Влучити в область</h1>

<p>Керуйте точкою за допомогою кнопок, щоб потрапити в цільову область.</p>

<canvas id="gameCanvas" width="400" height="400"></canvas>

<div class="controls">

<button onclick="movePoint(-10, 0)">⬅️ Вліво</button>

<button onclick="movePoint(10, 0)">➡️ Вправо</button>

<button onclick="movePoint(0, -10)">⬆️ Вверх</button>

<button onclick="movePoint(0, 10)">⬇️ Вниз</button>

</div>

<div id="hints">

<strong>Історія спроб:</strong>

<ul id="hintList"></ul>

</div>

<script>

const canvas = document.getElementById('gameCanvas');

const ctx = canvas.getContext('2d');

const hintList = document.getElementById('hintList');

// Опис області

const verticalStrip1 = { xStart: 50, xEnd: 150 }; // Фігура-1

const horizontalStrip = { yStart: 100, yEnd: 200 }; // Фігура-2

const verticalStrip2 = { xStart: 100, xEnd: 300 }; // Фігура-3

let pointX = Math.floor(Math.random() \* 400); // Початкова позиція точки

let pointY = Math.floor(Math.random() \* 400);

let previousDistance = null;

// Функція малювання області

function drawTargetArea() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

// Фігура-1: Перша вертикальна стрічка

ctx.fillStyle = 'rgba(255, 0, 0, 0.3)';

ctx.fillRect(verticalStrip1.xStart, 0, verticalStrip1.xEnd - verticalStrip1.xStart, canvas.height);

// Фігура-2: Горизонтальна стрічка

ctx.fillStyle = 'rgba(0, 255, 0, 0.3)';

ctx.fillRect(0, horizontalStrip.yStart, canvas.width, horizontalStrip.yEnd - horizontalStrip.yStart);

// Фігура-3: Друга вертикальна стрічка

ctx.fillStyle = 'rgba(0, 0, 255, 0.3)';

ctx.fillRect(verticalStrip2.xStart, 0, verticalStrip2.xEnd - verticalStrip2.xStart, canvas.height);

}

// Малюємо точку

function drawPoint() {

ctx.beginPath();

ctx.arc(pointX, pointY, 5, 0, Math.PI \* 2);

ctx.fillStyle = 'red';

ctx.fill();

ctx.stroke();

}

// Оновлення canvas

function updateCanvas() {

drawTargetArea();

drawPoint();

}

// Обчислення відстані до центра області

function calculateDistance(x, y) {

const centerX = (verticalStrip2.xStart + verticalStrip2.xEnd) / 2;

const centerY = (horizontalStrip.yStart + horizontalStrip.yEnd) / 2;

return Math.sqrt((x - centerX) \*\* 2 + (y - centerY) \*\* 2);

}

// Перевірка чи точка в області

function isPointInTargetArea(x, y) {

const inVertical1 = x >= verticalStrip1.xStart && x <= verticalStrip1.xEnd;

const inHorizontal = y >= horizontalStrip.yStart && y <= horizontalStrip.yEnd;

const inVertical2 = x >= verticalStrip2.xStart && x <= verticalStrip2.xEnd;

return (inVertical1 || inHorizontal) && inVertical2;

}

// Додавання підказки

function addHint(hint) {

const li = document.createElement('li');

li.textContent = hint;

hintList.appendChild(li);

}

// Переміщення точки

function movePoint(dx, dy) {

const prevX = pointX;

const prevY = pointY;

// Оновлення координат

pointX += dx;

pointY += dy;

const distance = calculateDistance(pointX, pointY);

if (isPointInTargetArea(pointX, pointY)) {

addHint(`🎯 Влучили в ціль на координатах (${pointX}, ${pointY})!`);

updateCanvas();

return;

}

if (previousDistance === null) {

addHint(`Початкові координати: (${pointX}, ${pointY}).`);

} else if (distance < previousDistance) {

addHint(`Тепліше: (${pointX}, ${pointY}), відстань до цілі зменшилася.`);

} else {

addHint(`Холодніше: (${pointX}, ${pointY}), відстань до цілі збільшилася.`);

}

previousDistance = distance;

// Оновлення canvas

updateCanvas();

}

// Початкове малювання

updateCanvas();

addHint(`Початкові координати точки: (${pointX}, ${pointY}).`);

</script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Обробка послідовності</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 20px;

}

#sequence {

display: flex;

justify-content: center;

flex-wrap: wrap;

margin-top: 20px;

}

.sequence-item {

margin: 5px;

padding: 10px 20px;

background-color: #f0f0f0;

border: 1px solid #ccc;

cursor: pointer;

user-select: none;

}

button {

padding: 10px 20px;

font-size: 16px;

margin: 10px;

cursor: pointer;

}

</style>

</head>

<body>

<h1>Обробка послідовності</h1>

<p>Оберіть файл для завантаження послідовності:</p>

<input type="file" id="inputFile" accept=".txt">

<button onclick="loadSequence()">Завантажити послідовність</button>

<div id="sequence"></div>

<div>

<button onclick="sortSequence()">Упорядкувати</button>

<button onclick="saveSequence()">Зберегти у файл</button>

</div>

<p>Оберіть місце для збереження результату:</p>

<input type="file" id="outputFile" accept=".txt" style="display:none;">

<p id="instructions">Натисніть на два елементи, щоб поміняти їх місцями.</p>

<script>

let sequence = [];

let firstSelectedIndex = null;

// Завантаження послідовності з файлу

function loadSequence() {

const inputFile = document.getElementById('inputFile').files[0];

if (!inputFile) {

alert('Оберіть файл для завантаження!');

return;

}

const reader = new FileReader();

reader.onload = function (e) {

sequence = e.target.result.split(/\s+/).filter(item => item.trim() !== '');

displaySequence();

};

reader.readAsText(inputFile);

}

// Відображення послідовності

function displaySequence() {

const sequenceDiv = document.getElementById('sequence');

sequenceDiv.innerHTML = '';

sequence.forEach((item, index) => {

const div = document.createElement('div');

div.className = 'sequence-item';

div.textContent = item;

div.setAttribute('data-index', index);

div.onclick = () => handleSwap(index);

sequenceDiv.appendChild(div);

});

}

// Обробка кліків для перестановки елементів

function handleSwap(index) {

if (firstSelectedIndex === null) {

firstSelectedIndex = index;

document.querySelector(`[data-index="${index}"]`).style.backgroundColor = 'yellow';

} else {

[sequence[firstSelectedIndex], sequence[index]] = [sequence[index], sequence[firstSelectedIndex]];

firstSelectedIndex = null;

displaySequence();

}

}

// Сортування послідовності

function sortSequence() {

sequence.sort((a, b) => (a > b ? 1 : -1));

displaySequence();

}

// Збереження послідовності у файл

function saveSequence() {

const outputFile = document.createElement('a');

const blob = new Blob([sequence.join(' ')], { type: 'text/plain' });

outputFile.href = URL.createObjectURL(blob);

outputFile.download = 'sorted\_sequence.txt';

outputFile.click();

}

</script>

</body>

</html>

4. Опис інтерфейсу програми (керівництво користувача)

**Завдання 1:**

1. Заголовок:  
   У верхній частині інтерфейсу розташований заголовок "Перестановка кнопок", який чітко вказує на основну функцію додатку — зміну місцями кнопок.
2. Інструкція:  
   Під заголовком знаходиться текстове повідомлення:  
   "Натисніть дві кнопки, щоб поміняти їх місцями."  
   Це просте пояснення того, як користувач може взаємодіяти з елементами на сторінці.
3. Кнопки:  
   Нижче інструкцій на сторінці відображаються п’ять кнопок з літерами A, B, C, D, E. Кожна кнопка має простий стиль і виглядає однаково. Вони розташовані горизонтально на сторінці.
   * Кожна кнопка є окремим елементом, на який можна натискати.
   * Кнопки мають однаковий стиль, із чіткими прямокутними межами і простим дизайном.
   * Після натискання на одну з кнопок, вона тимчасово зберігає вибір користувача (першу або другу кнопку для перестановки).
4. Взаємодія з інтерфейсом:  
   Користувач може натискати на будь-які дві кнопки, і вони будуть поміняні місцями. Після першого натискання, інтерфейс запам'ятовує вибрану кнопку (перша кнопка), а після другого натискання — виконуються перестановка обраних кнопок.
5. Колір та відображення:
   * Інтерфейс мінімалістичний, з простим білим фоном, що дозволяє акцентувати увагу на кнопках.
   * Кнопки мають легкий фоновий колір для виділення та інтуїтивно зрозуміле розташування.
6. Повідомлення про виконану дію:  
   Після кожної перестановки, внизу сторінки виводиться текст, який інформує користувача про те, які кнопки були поміняні місцями.

Інтерфейс є простим і зручним для користувача, орієнтуючись на наочність та простоту у взаємодії з елементами.

**Завдання 2:**

1. Заголовок: В верхній частині екрану знаходиться заголовок "Влучити в область". Це чітко інформує користувача про мету гри — переміщувати точку в цільову область.
2. Інструкція: Під заголовком є текст "Керуйте точкою за допомогою кнопок, щоб потрапити в цільову область". Це вказівка на основне завдання гри — перемістити точку до вказаної цільової області.
3. Поле гри (Canvas): У центрі екрану розташовано поле гри, яке є прямокутним малюнком із чіткими кольоровими секціями. Поле поділено на кілька кольорових зон:
   * Червона точка позначає поточне місце, на якому знаходиться точка, яку керує користувач.
   * Цільова область визначена певним кольором, і точка повинна потрапити в цю область.

Колірні ділянки на полі мають різні відтінки (наприклад, зелена, блакитна, фіолетова тощо), які допомагають орієнтуватися на полі.

1. Кнопки для керування: Нижче поля гри розташовані чотири кнопки:
   * Вліво — для переміщення точки вліво.
   * Вправо — для переміщення точки вправо.
   * Вверх — для переміщення точки вгору.
   * Вниз — для переміщення точки вниз.

Користувач може натискати на ці кнопки, щоб змінити координати точки і намагатися потрапити в цільову область.

1. Історія спроб: Під кнопками відображається блок "Історія спроб", де зберігаються попередні дії користувача. Наприклад, вказується початкові координати точки. Це дає користувачу можливість побачити прогрес у грі.
2. Візуальні ефекти: Коли точка переміщається, екран оновлюється, і точка відображається на нових координатах, що дозволяє гравцю відслідковувати свої рухи.

Інтерфейс гри мінімалістичний, що дозволяє зосередитись на головному завданні — переміщати точку в межах заданої області за допомогою кнопок.

**Завдання 3:**

1. **Заголовок:** В верхній частині екрану розташовано заголовок **"Обробка послідовності"**. Це вказує на основне завдання інтерфейсу — завантаження, обробка та збереження послідовностей.
2. **Керування файлами:**
   * **Кнопка "Виберіть файл"**: Користувач може натискати цю кнопку для вибору файлу з послідовністю, яку потрібно завантажити. Після вибору файлу буде виведено повідомлення про те, що файл обрано.
   * **Кнопка "Завантажити послідовність"**: Після вибору файлу, натискання цієї кнопки активує процес завантаження послідовності, що зберігається в текстовому файлі.
3. **Управління послідовністю:**
   * **Кнопка "Упорядкувати"**: Ця кнопка дає змогу користувачу упорядкувати елементи послідовності в алфавітному порядку.
   * **Кнопка "Зберегти у файл"**: Після редагування або упорядкування послідовності, натискання цієї кнопки дозволяє зберегти результат у новому файлі.
4. **Збереження результату:** Під кнопками для упорядкування та збереження знаходиться повідомлення **"Оберіть місце для збереження результату"**, що інформує користувача про необхідність вибору місця для збереження зміненої послідовності.
5. **Опис взаємодії з елементами:** Під інтерфейсними елементами є пояснення **"Натисніть на два елементи, щоб поміняти їх місцями"**. Це інструктує користувача, що для мануальної зміни місць елементів потрібно натискати на дві позиції в послідовності.

Інтерфейс простий і інтуїтивно зрозумілий, забезпечує функціонал для завантаження, обробки, перегляду та збереження результату.

5. Опис тестових прикладів

1. **Завдання 1: Перестановка кнопок**
   * **Мета**: Тестування функціональності перестановки місцями двох елементів при натисканні на кнопки.
   * **Тестові приклади**:
     1. **Вхідні дані**: Кнопки: A, B, C, D, E.  
        **Очікуваний результат**: Спочатку кнопки повинні бути в порядку: A, B, C, D, E.
        + Натискаємо на кнопку "A", потім на кнопку "E". Після цього кнопки повинні бути в порядку: E, B, C, D, A.
     2. **Вхідні дані**: Кнопки: X, Y, Z.  
        **Очікуваний результат**: Спочатку кнопки повинні бути в порядку: X, Y, Z.
        + Натискаємо на кнопку "X", потім на кнопку "Y". Після цього кнопки повинні бути в порядку: Y, X, Z.
     3. **Вхідні дані**: Кнопки: 1, 2, 3, 4.  
        **Очікуваний результат**: Спочатку кнопки повинні бути в порядку: 1, 2, 3, 4.
        + Натискаємо на кнопку "3", потім на кнопку "1". Після цього кнопки повинні бути в порядку: 3, 2, 1, 4.
2. **Завдання 2: Влучити в область**
   * **Мета**: Перевірка того, чи може користувач правильно орієнтувати точку та переміщати її за допомогою кнопок в межах певної області.
   * **Тестові приклади**:
     1. **Вхідні дані**: Початкові координати точки (256, 148).  
        **Очікуваний результат**: Точка повинна переміщатися в межах області відповідно до натискання кнопок "Вліво", "Вправо", "Верх", "Вниз". Користувач має перемістити точку всередину прямокутної області.
        + Натискаємо "Вправо" кілька разів, потім "Вниз". Точка повинна виявитися в середині цільової області.
     2. **Вхідні дані**: Початкові координати точки (100, 200).  
        **Очікуваний результат**: Точка повинна переміщуватися в межах області з урахуванням відповідних напрямків, вказаних користувачем.
        + Натискаємо "Вліво" та "Вверх". Точка повинна переміститися до верхнього лівого кута області.
     3. **Вхідні дані**: Початкові координати точки (350, 300).  
        **Очікуваний результат**: Точка має залишатися в межах області, навіть якщо користувач натискає на кнопки, що намагаються вивести точку за межі.
        + Користувач натискає "Вправо" та "Вниз", точка повинна переміститися в нижній правий кут або найближче можливе місце в межах області.
3. **Завдання 3: Обробка послідовності**
   * **Мета**: Тестування функціональності завантаження, упорядкування та збереження послідовності з файлу.
   * **Тестові приклади**:
     1. **Вхідні дані**: Файл з послідовністю: "яблуко банан грушка".  
        **Очікуваний результат**: Після завантаження та натискання кнопки "Упорядкувати", послідовність повинна стати: "банан грушка яблуко".
     2. **Вхідні дані**: Файл з послідовністю: "3 1 2".  
        **Очікуваний результат**: Після натискання кнопки "Упорядкувати", послідовність повинна стати: "1 2 3".
     3. **Вхідні дані**: Файл з послідовністю: "a b c".  
        **Очікуваний результат**: Після завантаження та натискання кнопки "Зберегти у файл", послідовність повинна зберегтися у новому текстовому файлі без змін, якщо не натискати кнопку "Упорядкувати".

Ці тестові приклади покривають базові функції кожного завдання і дозволяють перевірити коректність роботи кожної з них, враховуючи можливі варіанти введення та очікувані результати.

6. Аналіз помилок (опис усунення зауважень)

-