ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ

з дисципліни «Основи програмування на мові Javascript»

студента групи КС-21-2

Губа Максім

кафедра комп’ютерних технологій, ДНУ 2024/2025

**Постановка задачі:** 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. Вказати пару, яку треба переставити, шляхом натискання на елемент керування відобразити операнд перестановки (swap'а) в окремому місці, коли два операнда готові - своп по натисканню на окрему кнопку.

**Опис розв’язку:**

Для виконання перестановки значень змінних A, B, C, D, E у відповідний порядок AEDCB, спочатку програма приймає введення значень для кожної змінної. Щоб гарантувати коректне введення, реалізована перевірка на порожні значення. Якщо користувач не введе значення для змінної, програма повторно запитує введення.

Після того, як всі значення змінних отримано, їх зберігають у початковому стані для подальшого відображення. Використовуючи тимчасову змінну temp, здійснюється обмін значеннями між змінними згідно з правилами перестановки. Наприклад, значення змінної E перепризначається до змінної B, а значення D змінюється на значення C.

У кінці програма виводить повідомлення, в якому показані початкові та кінцеві значення змінних після перестановки.

**Вихідний текст програми розв’язку задачі**

// Створення елементів керування та перестановки  
function setupRearrangeUI() {  
 // Отримуємо контейнер для розміщення кнопок  
 const container = ***document***.createElement('div');  
 container.style.display = 'flex';  
 container.style.flexDirection = 'column';  
 container.style.alignItems = 'center';  
  
 // Заголовок  
 const title = ***document***.createElement('h3');  
 title.textContent = 'Переставте значення змінних у потрібному порядку';  
 title.style.width = '100%';  
 title.style.textAlign = 'center';  
 container.appendChild(title);  
  
 // Підконтейнер для кнопок змінних  
 const buttonContainer = ***document***.createElement('div');  
 buttonContainer.style.display = 'flex';  
 buttonContainer.style.flexDirection = 'row';  
 buttonContainer.style.justifyContent = 'center';  
 buttonContainer.style.flexWrap = 'wrap';  
 container.appendChild(buttonContainer);  
  
 // Масив змінних  
 const variables = ['A', 'B', 'C', 'D', 'E'];  
  
 // Зберігаємо стан змінних  
 const state = variables.map((variable) => ({ key: variable, value: prompt(`Введіть значення для ${variable}:`) }));  
  
 // Створення кнопок для змінних  
 const buttons = state.map((variable, index) => {  
 const button = ***document***.createElement('button');  
 button.textContent = `${variable.key}: ${variable.value}`;  
 button.style.margin = '5px';  
 button.style.padding = '10px';  
 button.dataset.index = index;  
 buttonContainer.appendChild(button);  
 return button;  
 });  
  
 // Додавання логіки swap  
 let selectedButton = null;  
 buttons.forEach(button => {  
 button.addEventListener('click', () => {  
 if (!selectedButton) {  
 selectedButton = button;  
 button.style.backgroundColor = 'lightblue';  
 } else {  
 // Виконуємо swap тільки значень  
 const firstIndex = selectedButton.dataset.index;  
 const secondIndex = button.dataset.index;  
  
 // Зміна значень у стані  
 [state[firstIndex].value, state[secondIndex].value] = [state[secondIndex].value, state[firstIndex].value];  
  
 // Оновлення тексту кнопок  
 buttons[firstIndex].textContent = `${state[firstIndex].key}: ${state[firstIndex].value}`;  
 buttons[secondIndex].textContent = `${state[secondIndex].key}: ${state[secondIndex].value}`;  
  
 // Скидання вибору  
 selectedButton.style.backgroundColor = '';  
 selectedButton = null;  
 }  
 });  
 });  
  
 // Додавання кнопки завершення  
 const finishButton = ***document***.createElement('button');  
 finishButton.textContent = 'Показати результат';  
 finishButton.style.marginTop = '20px';  
 container.appendChild(finishButton);  
 finishButton.addEventListener('click', () => {  
 const result = state.map(variable => `${variable.key}: ${variable.value}`).join(', ');  
 alert(`Результат перестановки: ${result}`);  
 });  
  
 // Додавання контейнера до документа  
 ***document***.body.appendChild(container);  
}  
  
// Ініціалізація програми  
setupRearrangeUI();

**Опис інтерфейсу програми (керівництво користувача):**

Після запуску програми користувачеві послідовно з'являються вікна із запитом на введення значень для змінних A, B, C, D, E.

У кожному вікні необхідно ввести відповідне значення для змінної. Якщо ввести порожнє значення, з'явиться повідомлення про помилку, і програма повторить запит.

Після успішного введення всіх значень, програма автоматично виконає перестановку.

Результат перестановки відображається у вигляді спливаючого повідомлення, яке містить початкові та змінені значення змінних.

**Опис тестових прикладів:**

Тест 1:

Вхід:

A = 1, B = 2, C = 3, D = 4, E = 5

Очікуваний результат:

A: 1, B: 3, C: 2, D: 3, Е: 4



Тест 2:

Вхід:

A = "a", B = "b", C = "c", D = "d", E = "e"

Очікуваний результат:

A: "a", B: "e", C: "c", D: "d", E: "b"

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Усі тестові приклади проходять без помилок та забезпечують правильний результат.

**Завдання 2**

**Постановка задачі**: Програма малює задану область на канві (canvas) веб-сторінки засобами мови JavaScript. Корекцію координат точки користувач здійснює шляхом натискання на кнопки "вліво", "вправо", "вверх", "вниз". Підказки користувачеві програма перезаписує і показує "відстань".

**Опис розв’язку:**

Задача полягає в тому, щоб відобразити трикутник відносно осі ординат (ось y). Для цього було реалізовано функцію, яка приймає координати трьох вершин трикутника і відображає їх. Відображення по осі y передбачає зміну знака координати x для кожної точки, при цьому координата y залишається незмінною.

Спочатку користувач вводить координати вершин трикутника через послідовні запити у вікні. Дані вводяться для кожної точки (A, B, C). Щоб уникнути некоректного введення, додана валідація значень на випадок, якщо користувач введе нечислові дані. Якщо введення не відповідає числовим значенням, програма повторює запит.

Після отримання початкових координат трикутника, вони передаються у функцію відображення, яка змінює знак x для кожної точки. Після виконання операції початкові та відображені координати виводяться на екран у вигляді повідомлення.

**Вихідний текст програми розв’язку задачі :**// Налаштування структури HTML  
const ***canvas*** = ***document***.createElement('canvas');  
***canvas***.width = 600;  
***canvas***.height = 600;  
***canvas***.style.border = '1px solid black';  
***document***.body.appendChild(***canvas***);  
  
const ***ctx*** = ***canvas***.getContext('2d');  
  
const ***instructions*** = ***document***.createElement('div');  
***instructions***.innerHTML = '<p>Використовуйте кнопки нижче, щоб перемістити точку. Спробуйте потрапити в цільову область!</p>';  
***document***.body.appendChild(***instructions***);  
  
const ***controls*** = ***document***.createElement('div');  
***controls***.innerHTML = `  
 <button id="left">Вліво</button>  
 <button id="right">Вправо</button>  
 <button id="up">Вгору</button>  
 <button id="down">Вниз</button>  
`;  
***document***.body.appendChild(***controls***);  
  
const ***feedback*** = ***document***.createElement('div');  
***feedback***.innerHTML = '<p>Відгук: </p>';  
***document***.body.appendChild(***feedback***);  
  
const ***history*** = ***document***.createElement('div');  
***history***.innerHTML = '<p>Історія:</p>';  
***document***.body.appendChild(***history***);  
  
// Визначення меж цільових областей  
const ***verticalStrip1*** = { x1: 100, x2: 200 }; // Перша вертикальна смужка  
const ***verticalStrip2*** = { x1: 400, x2: 500 }; // Друга вертикальна смужка  
const ***horizontalStrip*** = { y1: 300, y2: 400 }; // Горизонтальна смужка  
  
// Випадкова початкова точка для користувача  
let ***point*** = {   
 x: ***Math***.floor(***Math***.random() \* ***canvas***.width),   
 y: ***Math***.floor(***Math***.random() \* ***canvas***.height)  
};  
let ***prevDistance*** = null;  
let ***attempts*** = 0;  
  
// Малювання цільової області  
function drawTargetArea() {  
 ***ctx***.fillStyle = 'lightblue';  
  
 // Вертикальні смужки  
 ***ctx***.fillRect(***verticalStrip1***.x1, 0, ***verticalStrip1***.x2 - ***verticalStrip1***.x1, ***canvas***.height);  
 ***ctx***.fillRect(***verticalStrip2***.x1, 0, ***verticalStrip2***.x2 - ***verticalStrip2***.x1, ***canvas***.height);  
  
 // Горизонтальна смужка  
 ***ctx***.fillRect(0, ***horizontalStrip***.y1, ***canvas***.width, ***horizontalStrip***.y2 - ***horizontalStrip***.y1);  
}  
  
// Малювання точки  
function drawPoint() {  
 ***ctx***.fillStyle = 'red';  
 ***ctx***.beginPath();  
 ***ctx***.arc(***point***.x, ***point***.y, 5, 0, ***Math***.PI \* 2);  
 ***ctx***.fill();  
}  
  
// Оновлення канви  
function updateCanvas() {  
 ***ctx***.clearRect(0, 0, ***canvas***.width, ***canvas***.height);  
 drawTargetArea();  
 drawPoint();  
}  
  
// Перевірка, чи знаходиться точка в цільовій області  
function isInsideTargetArea(x, y) {  
 const inVertical1 = x >= ***verticalStrip1***.x1 && x <= ***verticalStrip1***.x2;  
 const inVertical2 = x >= ***verticalStrip2***.x1 && x <= ***verticalStrip2***.x2;  
 const inHorizontal = y >= ***horizontalStrip***.y1 && y <= ***horizontalStrip***.y2;  
 return (inVertical1 || inVertical2) && inHorizontal;  
}  
  
// Обробка переміщення користувача  
function movePoint(direction) {  
 const movement = 10;  
 if (direction === 'left') ***point***.x = ***Math***.max(0, ***point***.x - movement);  
 if (direction === 'right') ***point***.x = ***Math***.min(***canvas***.width, ***point***.x + movement);  
 if (direction === 'up') ***point***.y = ***Math***.max(0, ***point***.y - movement);  
 if (direction === 'down') ***point***.y = ***Math***.min(***canvas***.height, ***point***.y + movement);  
  
 ***attempts***++;  
  
 const distance = ***Math***.abs(***point***.x - 300) + ***Math***.abs(***point***.y - 350); // Центр цільової області  
  
 let feedbackText;  
 if (isInsideTargetArea(***point***.x, ***point***.y)) {  
 feedbackText = `Вітаємо! Ви потрапили в цільову область з ${***attempts***}-ї спроби.`;  
 disableControls();  
 } else {  
 if (***prevDistance*** === null) {  
 feedbackText = 'Промах. Спробуйте ще раз!';  
 } else if (distance < ***prevDistance***) {  
 feedbackText = 'Стає тепліше!';  
 } else {  
 feedbackText = 'Стає холодніше!';  
 }  
 ***prevDistance*** = distance;  
 }  
  
 ***feedback***.innerHTML = `<p>Відгук: ${feedbackText}</p>`;  
 ***history***.innerHTML += `<p>Спроба ${***attempts***}: (${***point***.x}, ${***point***.y}) - ${feedbackText}</p>`;  
 updateCanvas();  
}  
  
function disableControls() {  
 ***document***.getElementById('left').disabled = true;  
 ***document***.getElementById('right').disabled = true;  
 ***document***.getElementById('up').disabled = true;  
 ***document***.getElementById('down').disabled = true;  
}  
  
// Слухачі подій для кнопок  
***document***.getElementById('left').addEventListener('click', () => movePoint('left'));  
***document***.getElementById('right').addEventListener('click', () => movePoint('right'));  
***document***.getElementById('up').addEventListener('click', () => movePoint('up'));  
***document***.getElementById('down').addEventListener('click', () => movePoint('down'));  
  
// Початкова налаштування  
updateCanvas();  
***feedback***.innerHTML = `<p>Відгук: Початок у точці (${***point***.x}, ${***point***.y})</p>`;

**Опис інтерфейсу програми (керівництво користувача):**

1. Після запуску програми користувачеві з'являються запити на введення координат для трьох точок трикутника: A, B, C.
2. Для кожної точки необхідно ввести координати x і y. Програма перевіряє, щоб введені значення були числовими. Якщо значення не є числом, користувач отримає повідомлення про помилку і буде повторно запрошено ввести значення.
3. Після введення всіх координат, програма автоматично відобразить трикутник по осі y.
4. Результати відображення трикутника виводяться у вигляді спливаючого повідомлення, яке містить початкові та відображені координати точок трикутника.

**Опис тестових прикладів:**

**Тест 1:**

Изображение выглядит как текст, снимок экрана, Прямоугольник, графический дизайн

Автоматически созданное описание

Усі тестові приклади успішно виконуються без помилок.

**Аналіз помилок (опис усунення зауважень):**

Під час тестування програми було виявлено проблему з введенням некоректних значень. Наприклад, якщо користувач залишав поле для введення порожнім або вводив текст, програма видавала помилку і завершувала виконання. Для виправлення цієї проблеми була додана перевірка введених значень на наявність чисел. Якщо введене значення не є числом, програма повторює запит до користувача.

**Завдання 3**

**Постановка задачі:** У цьому завданні створено канвас, на якому можна переміщувати точку за допомогою клавіатури. На канвасі малюється початкова точка та фігури, такі як лінії та прямокутники. Для управління точкою використовуються обробники подій клавіатури, які реагують на натискання стрілок (вгору, вниз, вліво, вправо), і змінюють координати точки на канвасі. При кожному переміщенні точка повторно відображається в новій позиції. Додатково, якщо точка потрапляє в одну з фігур, з'являється повідомлення про це.

**Опис розв’язку:**

Алгоритм реалізований у вигляді функції, яка приймає координати точки та межі смужок, а потім перевіряє, чи потрапляють координати точки у межі кожної з цих смужок. Якщо точка знаходиться в одній або обох смужках, вона вважається такою, що потрапляє у об'єднання фігур.

Для уникнення помилок користувачеві надається можливість повторно вводити дані у випадку, якщо було введено некоректні значення (нечислові значення). Програма викликає додаткову перевірку та повторно запитує введення до тих пір, поки не буде отримано коректні значення.

**Вихідний текст програми розв’язку задачі :  
JS**  
let ***numbers*** = [];  
let ***selectedIndices*** = [];  
  
async function uploadFile() {  
 const fileInput = ***document***.getElementById('fileInput');  
 if (fileInput.files.length === 0) {  
 alert('Будь ласка, виберіть файл.');  
 return;  
 }  
  
 const file = fileInput.files[0];  
 if (file.type && file.type !== 'text/plain') {  
 alert('Помилка: виберіть текстовий файл (.txt).');  
 return;  
 }  
  
 try {  
 const reader = new FileReader();  
 reader.onload = function(event) {  
 const text = event.target.result;  
  
 // Очищення тексту від зайвих символів і роздільників  
 const cleanedText = text.replace(/[^\d\s-]/g, ''); // Дозволені тільки цифри, пробіли та мінус  
 const parsedNumbers = cleanedText  
 .split(/[\s,\n]+/) // Розділення за пробілами, комами або новими рядками  
 .map(num => Number(num)) // Перетворення у числа  
 .filter(num => !isNaN(num)); // Відфільтровування нечислових значень  
  
 if (parsedNumbers.length === 0) {  
 alert('Помилка: файл не містить дійсних чисел.');  
 return;  
 }  
  
 ***numbers*** = parsedNumbers;  
 displayNumbers();  
 ***document***.getElementById('output').innerText = `Завантажені числа: ${***numbers***.join(', ')}`;  
 };  
 reader.readAsText(file);  
 } catch (error) {  
 alert('Помилка читання файлу. Переконайтеся, що це текстовий файл.');  
 }  
}  
  
function downloadFile() {  
 if (***numbers***.length === 0) {  
 alert('Немає даних для завантаження. Спочатку завантажте файл.');  
 return;  
 }  
 const blob = new Blob([***numbers***.join(' ')], { type: 'text/plain' });  
 const a = ***document***.createElement('a');  
 a.href = ***URL***.createObjectURL(blob);  
 a.download = 'sorted\_numbers.txt';  
 a.click();  
}  
  
function displayNumbers() {  
 const container = ***document***.getElementById('numberContainer');  
 container.innerHTML = '';  
 ***numbers***.forEach((num, index) => {  
 const div = ***document***.createElement('div');  
 div.classList.add('number');  
 div.textContent = num;  
 div.onclick = () => handleNumberClick(index);  
 if (***selectedIndices***.includes(index)) {  
 div.classList.add('selected');  
 }  
 container.appendChild(div);  
 });  
}  
  
function handleNumberClick(index) {  
 if (!***selectedIndices***.includes(index)) {  
 ***selectedIndices***.push(index);  
 } else {  
 ***selectedIndices*** = ***selectedIndices***.filter(i => i !== index);  
 }  
  
 if (***selectedIndices***.length === 2) {  
 swapNumbers(***selectedIndices***[0], ***selectedIndices***[1]);  
 ***selectedIndices*** = [];  
 }  
 displayNumbers();  
}  
  
function swapNumbers(index1, index2) {  
 [***numbers***[index1], ***numbers***[index2]] = [***numbers***[index2], ***numbers***[index1]];  
 ***document***.getElementById('output').innerText = `Переставлено ${index1} і ${index2}: ${***numbers***.join(', ')}`;  
}  
  
function digitCount(n) {  
 return ***Math***.abs(n).toString().length;  
}  
  
function sortNumbers() {  
 if (***numbers***.length === 0) {  
 alert('Будь ласка, спочатку завантажте файл!');  
 return;  
 }  
 for (let i = 0; i < ***numbers***.length - 1; i++) {  
 let minIndex = i;  
 for (let j = i + 1; j < ***numbers***.length; j++) {  
 if (digitCount(***numbers***[j]) < digitCount(***numbers***[minIndex])) {  
 minIndex = j;  
 }  
 }  
 [***numbers***[i], ***numbers***[minIndex]] = [***numbers***[minIndex], ***numbers***[i]];  
 }  
 ***document***.getElementById('output').innerText = `Відсортовані числа: ${***numbers***.join(', ')}`;  
 displayNumbers();  
}

HTML  
<!-- index.html -->  
<!DOCTYPE html>  
<html lang="uk">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Інтерфейс сортування</title>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
 <h1>Інтерфейс сортування</h1>  
 <div class="container">  
 <div class="row">  
 <input type="file" id="fileInput" accept=".txt" />  
 <button type="button" onclick="uploadFile()">Завантажити</button>  
 </div>  
 <div class="row">  
 <button type="button" onclick="downloadFile()">Завантажити файл</button>  
 <button type="button" onclick="sortNumbers()">Сортувати за кількістю цифр</button>  
 </div>  
 <div id="output">Зміст файлу з'явиться тут.</div>  
 <div id="numberContainer" class="number-container"></div>  
 </div>  
 <script src="script.js"></script>  
</body>  
</html>

**Опис інтерфейсу програми (керівництво користувача):**

1. Програма послідовно запитує у користувача параметри для вертикальної та горизонтальної смужок:
   * Мінімальну та максимальну координати по осі x для вертикальної смужки.
   * Мінімальну та максимальну координати по осі y для горизонтальної смужки.
2. Після цього користувач вводить координати точки (x, y), для якої необхідно перевірити потрапляння у смужки.
3. Програма перевіряє, чи точка знаходиться у межах заданих смужок, і виводить відповідне повідомлення в консоль:
   * Якщо точка знаходиться хоча б у одній із смужок, програма виводить повідомлення про те, що точка потрапляє у об'єднання смужок.
   * Якщо точка не потрапляє у жодну смужку, виводиться відповідне повідомлення.
4. Якщо при введенні даних користувач вводить некоректне значення (наприклад, нечислові символи), програма виводить помилку і запитує повторне введення.

**Опис тестових прикладів:**

**Тест 1:**  
Вхідні дані:  
23 45 67

89 12 34

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**

**Аналіз помилок (опис усунення зауважень):**

Під час проведення тесту виникла проблема з завантаженням та збереженням файлу, котру було виправлено в кінцевому результаті   
Також були проведені додаткові тести для перевірки точності роботи програми при введенні від’ємних значень або значень, що знаходяться на межі смужок. Програма коректно розпізнає випадки, коли точка потрапляє на межі або поза межами смужок.