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

**Завдання 1**

**Постановка задачі:**

Необхідно реалізувати програму, яка дозволяє користувачеві "влучити" точкою в область на площині, утворену об'єднанням двох фігур (вертикальної смужки та першої горизонтальної смужки), що перетинається з третьою фігурою (другою горизонтальною смужкою). Користувач має кілька спроб ввести координати точки, а програма повинна супроводжувати спроби підказками про те, чи наближається точка до мети: "тепліше" або "холодніше". Коли користувач влучить в область, програма повинна вивести повідомлення з кількістю спроб, за яку це сталося.

**Опис розв’язку:**

Програма приймає координати горизонтальної смужки та двох вертикальних смужок, а також координати точки, в яку користувач намагається "влучити". Перевіряється, чи потрапляє точка в одну з двох фігур: горизонтальну смужку або одну з вертикальних смужок. Об'єднання цих фігур утворює область, де точка вважається "влученою", якщо вона потрапляє одночасно в обидва вертикальні діапазони. Користувач може вводити координати точки до тих пір, поки вона не опиниться в межах цієї області. Програма також оцінює кожну спробу: якщо точка наближається до центру області, виводиться повідомлення "тепліше", якщо віддаляється — "холодніше".

**Оновлений алгоритм:**

1. Програма запитує у користувача координати для однієї вертикальної та двох горизонтальних смужок.
2. Користувач вводить координати точки.
3. Програма перевіряє, чи потрапляє точка в одну або кілька заданих смужок.
4. Якщо точка потрапила в область, утворену об'єднанням смужок, програма повідомляє про успішне влучення.
5. Якщо точка не потрапила в область, програма обчислює відстань до центра цільової області та виводить підказку: "тепліше" або "холодніше", залежно від того, наближається чи віддаляється точка.
6. Користувач має обмежену кількість спроб, щоб ввести правильні
7. координати.

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

1. Користувач вводить координати точки.
2. Програма перевіряє, чи потрапляє точка в задану область.
3. Якщо точка потрапляє в область, виводиться повідомлення про успіх, із зазначенням кількості спроб.
4. Якщо точка не потрапляє в область, програма підказує користувачеві, чи наближається він до мети.

**Вихідний текст програми розв’язку задачі :**const readline = require('readline');  
  
// Определяем границы фигур  
const verticalStrip1 = { x1: 1, x2: 3 }; // Первая вертикальная полоска  
const verticalStrip2 = { x1: 7, x2: 9 }; // Вторая вертикальная полоска  
const horizontalStrip = { y1: 4, y2: 6 }; // Горизонтальная полоска  
  
// Проверяем, попадает ли точка в целевую область  
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;  
}  
  
const rl = readline.createInterface({  
 input: process.stdin,  
 output: process.stdout  
});  
  
let prevDistance = null;  
let attempts = 0;  
  
function askUser() {  
 rl.question('Введите координаты x и y через пробел: ', (input) => {  
 const [x, y] = input.split(' ').map(Number);  
 if (isNaN(x) || isNaN(y)) {  
 ***console***.log('Введите корректные числа.');  
 return askUser();  
 }  
 attempts++;  
 const distance = ***Math***.abs(x - 5) + ***Math***.abs(y - 5); // Расстояние от центра области  
  
 if (isInsideTargetArea(x, y)) {  
 ***console***.log(`Поздравляем! Вы попали в область с ${attempts}-й попытки.`);  
 rl.close();  
 } else {  
 if (prevDistance === null) {  
 ***console***.log('Промах. Попробуйте снова.');  
 } else if (distance < prevDistance) {  
 ***console***.log('Теплее.');  
 } else {  
 ***console***.log('Холоднее.');  
 }  
 prevDistance = distance;  
 askUser();  
 }  
 });  
}  
  
***console***.log('Попробуйте попасть в целевую область.');  
askUser();

**Опис тестових прикладів:**

Тест 1:

Изображение выглядит как текст, снимок экрана, Шрифт, Цвет электрик

Автоматически созданное описание

**Аналіз помилок та опис їх усунення:**

Основними проблемами були некоректні введення, зокрема нечислові значення або порожні поля. Для вирішення цієї проблеми в програму була додана валідація введених даних, яка перевіряє коректність формату та виводить повідомлення про помилку у випадку некоректного вводу.

Також було проведено тестування для наступних сценаріїв:

1. **Від'ємні значення координат:** Перевірено, як програма реагує на введення точок із від'ємними значеннями. Тестування підтвердило правильність роботи алгоритму.
2. **Координати на межах смуг:** Виконані перевірки для точок, що лежать на границях фігур, щоб упевнитися у точності логіки перевірки.

Завдяки цим покращенням програма стала більш стабільною, надійною та здатною обробляти різні випадки введення даних.

**Задача 2**

**Постановка задачі:**

Задано послідовність цілих чисел. Напишіть програму, яка за один (за можливості) перегляд послідовності, для елементів, що задовольняють заданому критерію, обчислює задану величину. Критерій і шукана величина визначені індивідуальним варіантом. Не використовуйте масиви, якщо в цьому немає потреби.

**Опис розв'язку:**

Програма повинна послідовно отримувати числа від користувача і на основі заданого критерію визначити індекс останнього найбільшого числа, яке має старшу цифру, відмінну від 9. Для цього програма перевіряє кожне введене число та порівнює його з поточним найбільшим числом. Якщо це число більше за поточне найбільше, воно стає новим найбільшим і його індекс зберігається. Якщо воно рівне найбільшому, індекс також оновлюється.

Крім того, програма повинна перевіряти, що старша цифра числа не дорівнює 9, і лише в цьому випадку враховувати число.

Процес завершиться, коли користувач введе команду 'end'. Після завершення програма виведе індекс останнього найбільшого числа, яке задовольняє умови.

**Алгоритм:**

1. **Сума власних дільників числа:**  
   Для обчислення суми власних дільників числа використовується функція sumOfDivisors. Вона перевіряє всі числа від 1 до n-1 на дільність числа n і додає ці числа до суми.
2. **Один прохід через послідовність:**  
   У циклі, який читає числа послідовності:

* Підтримуємо мінімальне значення (minValue).
* Якщо зустрічається нове менше число, оновлюємо minValue і обнуляємо лічильник (count).
* Якщо число дорівнює поточному мінімуму, перевіряємо, чи парна сума його власних дільників, і, якщо так, збільшуємо лічильник.

1. **Умова зупинки:**  
   Програма завершує роботу після одного проходу через послідовність, одразу отримуючи кінцевий результат.

**Текст програми:**const readline = require('readline');  
  
// Функция для нахождения суммы собственных делителей числа  
function sumOfDivisors(n) {  
 let sum = 0;  
 for (let i = 1; i < n; i++) {  
 if (n % i === 0) sum += i;  
 }  
 return sum;  
}  
  
const rl = readline.createInterface({  
 input: process.stdin,  
 output: process.stdout  
});  
  
rl.question('Введите последовательность чисел через пробел: ', (input) => {  
 const numbers = input.split(' ').map(Number);  
 if (numbers.some(isNaN)) {  
 ***console***.log('Введите корректные числа.');  
 rl.close();  
 return;  
 }  
  
 let minValue = ***Infinity***;  
 let count = 0;  
  
 for (const num of numbers) {  
 if (num <= 0) continue; // Пропускаем нули и отрицательные числа  
  
 const sumDiv = sumOfDivisors(num);  
  
 // Відлагоджувальний лог  
 ***console***.log(`Число: ${num}, Сума дільників: ${sumDiv}, Мінімальне значення: ${minValue}`);  
  
 // Якщо число менше поточного мінімального  
 if (num < minValue) {  
 minValue = num;  
 count = (sumDiv % 2 === 0) ? 1 : 0; // Скидаємо лічильник і перевіряємо парність  
 } else if (num === minValue && sumDiv % 2 === 0) {  
 count++; // Збільшуємо лічильник, якщо число дорівнює мінімальному і має парну суму дільників  
 }  
 }  
  
 ***console***.log(`Количество минимальных значений с чётной суммой собственных делителей: ${count}`);  
 rl.close();  
});

**Тестові приклади:**

**Тест 1:**

**Вхідні дані:**

**Послідовність: 4, 5, 6, 7, 8, 9**

1. **Знаходимо мінімальне число:**
   * Мінімальне число в послідовності: 4.
2. **Обчислюємо суму власних дільників кожного числа:**
   * Для 4: власні дільники — 1, 2. Сума = 1 + 2 = 3 (непарна).
   * Для 5: власні дільники — 1. Сума = 1 (непарна).
   * Для 6: власні дільники — 1, 2, 3. Сума = 1 + 2 + 3 = 6 (парна).
   * Для 7: власні дільники — 1. Сума = 1 (непарна).
   * Для 8: власні дільники — 1, 2, 4. Сума = 1 + 2 + 4 = 7 (непарна).
   * Для 9: власні дільники — 1, 3. Сума = 1 + 3 = 4 (парна).
3. **Перевіряємо, скільки разів зустрічається мінімальне число (4) із парною сумою власних дільників:**
   * Мінімальне число — 4. Сума його власних дільників = 3 (непарна).

**Висновок:**



**Тест 2:**

**Вхідні дані:**

* Числа: 4 2 8 12 20

**Очікуваний результат:** Число, що задовольняє умови, не знайдено.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Тест 3:**

**Вхідні дані:**

* Числа: 10, 20, 10, 20
* Ключове значення: індекс останнього найбільшого числа, старша цифра якого не 9.

**Очікуваний результат:** Індекс останнього найбільшого числа: 3 (число 20).

**Пояснення:**

* Перше число: 10, старша цифра не 9, стає найбільшим, індекс 0.
* Друге число: 20, старша цифра не 9, нове найбільше число, індекс 1.
* Третє число: 10, старша цифра не 9, не є більшим за поточне найбільше 20, індекс залишається 1.
* Четверте число: 20, старша цифра не 9, рівне поточному найбільшому, індекс оновлюється на 3.
* Після введення 'end', програма виводить індекс останнього найбільшого числа — 3 (число 20).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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

1. **Врахування нульових значень:**  
   Помилка виникала через обробку числа 0, яке неправильно враховувалося як мінімальне значення. Це стало причиною хибного підрахунку, оскільки сума власних дільників для нуля дорівнює 0 (парна).  
   **Виправлення:** Додано перевірку num > 0, щоб виключити нуль та від'ємні числа з обчислень.
2. **Ініціалізація мінімального значення:**  
   Використання Infinity як початкового значення для змінної minValue є правильним, але потребувало уточнення логіки оновлення.  
   **Виправлення:** Перевірка мінімального значення тепер здійснюється лише для додатних чисел.
3. **Перевірка парності суми дільників:**  
   Логіка перевірки парності суми дільників залишалася правильною, але через неправильне врахування нуля результати були некоректними.  
   **Виправлення:** Після виключення нуля з обчислень підрахунок мінімальних значень із парною сумою власних дільників працює коректно.
4. **Додаткові логічні перевірки:**  
   Додані умови дозволили уникнути помилок при обробці вхідних даних, що забезпечує стабільність програми.

**Завдання 3**

**Постановка задачі**

Напиcати програму, яка вводить послідовність цілих чисел a[0], a[1], ... , a[N-1] і упорядковує її згідно до варіанту завдання певним методом за певним критерієм у певному напрямку. Передбачте можливість файлового вводу і виводу даних за допомогою перенапрямлення вводу і виводу в командному рядку. Метод упорядкування реалізуйте у вигляді окремої функції (стандартні функції упорядкування використовувати не дозволяється).

**Алгоритм**

1. Вхідні дані:  
   Отримати послідовність цілих чисел як вхідний масив a[0], a[1], ..., a[N-1].
2. Підготовка індексів:  
   Створити масив індексів indices, де кожен елемент відповідає позиції числа у вхідному масиві:

* Наприклад, для a = [2413, 12, 7, 12345, 456] індекси будуть [0, 1, 2, 3, 4].

1. Підрахунок кількості цифр:  
   Реалізувати функцію digitCount(n), яка повертає кількість цифр у числі n:

* Конвертувати число в його абсолютне значення (щоб обробляти від'ємні числа).
* Перетворити число в рядок і порахувати довжину рядка.

1. Сортування вибором (Selection Sort):

* Порівнювати кількість цифр чисел на відповідних індексах.
* Сортувати масив індексів indices так, щоб кількість цифр у числах йшла у порядку неспадання.

Алгоритм сортування вибором:

* Для кожної позиції i від 0 до N-2:
  1. Знайти індекс minIndex, на якому число має мінімальну кількість цифр серед усіх чисел від i до N-1.
  2. Обміняти місцями indices[i] та indices[minIndex].
* Повторювати, поки весь масив індексів не буде відсортовано.

1. Вивід результату:  
   Вивести відсортований масив індексів.

**Вихідний текст програми розв’язку задачі**

const readline = require('readline');  
  
// Функция для подсчёта цифр в числе  
function digitCount(n) {  
 return ***Math***.abs(n).toString().length;  
}  
  
// Сортировка выбором (Selection Sort)  
function selectionSortByDigitCount(arr) {  
 const indices = arr.map((\_, i) => i);  
 for (let i = 0; i < indices.length - 1; i++) {  
 let minIndex = i;  
 for (let j = i + 1; j < indices.length; j++) {  
 if (digitCount(arr[indices[j]]) < digitCount(arr[indices[minIndex]])) {  
 minIndex = j;  
 }  
 }  
 [indices[i], indices[minIndex]] = [indices[minIndex], indices[i]];  
 }  
 return indices;  
}  
  
// Создание интерфейса для ввода  
const rl = readline.createInterface({  
 input: process.stdin,  
 output: process.stdout  
});  
  
rl.question('Введите последовательность чисел через пробел: ', (input) => {  
 const numbers = input.split(' ').map(Number);  
 if (numbers.some(isNaN)) {  
 ***console***.log('Ошибка: введите корректные числа.');  
 rl.close();  
 return;  
 }  
  
 const sortedIndices = selectionSortByDigitCount(numbers);  
 ***console***.log(`Отсортированные индексы: ${sortedIndices.join(' ')}`);  
 rl.close();  
});

**Результат**

* Число 7 (1 цифра) → индекс 2.
* Число 12 (2 цифры) → индекс 1.
* Число 456 (3 цифры) → индекс 4.
* Число 2413 (4 цифры) → индекс 0.
* Число 12345 (5 цифр) → индекс 3.  
  Изображение выглядит как текст, снимок экрана, Шрифт, Цвет электрик

  Автоматически созданное описание

**Аналіз помилок**

1. **Некоректний ввід:**
   * Якщо користувач вводить некоректні дані (наприклад, текст), програма ігнорує це введення і запитує нове.
2. **Порожній ввід:**
   * Якщо послідовність чисел відсутня, програма не виконує сортування і завершує роботу.

**Висновок**

У результаті виконання даної лабораторної роботи було розроблено три програми на **Node.js**, які забезпечують інтерактивний діалог із користувачем у консолі операційної системи. В процесі роботи я ознайомився з основами роботи з модулем **readline**, який використовується для зчитування вводу з консолі, а також із методами обробки даних, що передаються користувачем. У кожній програмі реалізовано функції для організації вводу, обробки та виведення результатів у зрозумілому для користувача форматі. Для розв'язання задачі впорядкування індексів за кількістю цифр у числах було застосовано алгоритм **сортування вибором**.