Лабораторна робота №3
Робота з одновимірними та багатовимірними масивами
Мета: навчитися створювати і обробляти одновимірні та багатовимірні масиви та знайомство з операціями над ними.
Хід виконання

Завдання 1.

1.1. Розмістіть на сторінці **div**-блок з ідентифікатором **“paragraphs”**. Розмістіть у ньому 5 абзаців з довільним текстом. Сформуйте масив, який міститиме 5 елементів - кількість символів, які входять до відповідного абзацу тексту.
Для пошуку абзаців використовуйте метод **querySelectorAll** та відповідний CSS-селектор. Для доступу до тексту, який входить в абзац використовуйте властивість **innerText** для DOM-елемента. Виведіть у консоль отриманий масив.

1.2. Дано масив цілих чисел:

let arr = [20, 17, 4, -4, 10, -9, 13, 4, 12, 22, 13, 19, 1, 3];

- Для заданого масиву визначте та виведіть у консоль:
  - найменше та найбільше значення;
  - кількість парних та непарних чисел;
  - кількість чисел, що містять дві цифри;
  - кількість додатних та кількість від’ємних чисел.
- Відсортуйте масив **arr** за допомогою методу Sort у порядку зростання значень та виведіть у консоль результат.
- Створіть масив з оберненим порядком елементів та виведіть його у консоль.

Лістинг index.html:

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
    <title>Task 01</title>
</head>
<body>
<div id="paragraphs">
    <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard </p>
    <p>dummy text ever since the 1500s, when an unknown printer took </p>
    <p>a galley of type and scrambled it to make a type specimen book. </p>
    <p> It has survived not only five centuries, but also the leap into electronic typesetting</p>
```

```
    <p>PageMaker including versions of Lorem Ipsum</p>
</div>
<script>
    let div = document.getElementById('paragraphs');
    let array = [];
    let paragraphs = div.querySelectorAll('p');

    paragraphs.forEach(p => {
        array.push(p.innerText.length);
    });

    console.log(array);

    let arr = [20, 17, 4, -4, 10, -9, 13, 4, 12, 22, 13, 19, 1, 3];
    let min = arr[0];
    let max = arr[0];
    let plus_count = 0;
    let minus_count = 0;
    let parni_count = 0;
    let neparni_count = 0;
    let double_count = 0;

    for (let num of arr) {
        if (num < min) {
            min = num;
        }

        if (num > max) {
            max = num;
        }

        if (num > 0) {
            plus_count++;
        }

        if (num < 0) {
            minus_count++;
        }

        if (num % 2 === 0) {
            parni_count++;
        } else {
            neparni_count++;
        }

        if ((num >= 10 && num <= 99) || (num <= -10 && num >= -99)) {
            double_count++;
        }
    }

    console.log(arr);
    console.log(`min: ${min}`);
    console.log(`max: ${max}`);
    console.log(`plus_count: ${plus_count}`);
    console.log(`minus_count: ${minus_count}`);
    console.log(`parni_count: ${parni_count}`);
    console.log(`neparni_count: ${neparni_count}`);
    console.log(`double_count: ${double_count}`);

    arr.sort((a, b) => a - b);
    console.log(arr);
```

```
    let reversedArr = [...arr].reverse();
    console.log(reversedArr);
</script>
</body>
</html>
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard

dummy text ever since the 1500s, when an unknown printer took

a galley of type and scrambled it to make a type specimen book.

It has survived not only five centuries, but also the leap into electronic typesetting

PageMaker including versions of Lorem Ipsum

```
Inspector  Console  Debugger  Network  Style Editor  Performance  Memory  Storage  Acce
Filter Output
▶ Array(5) [ 119, 61, 63, 86, 43 ]
▶ Array(14) [ -9, -4, 1, 3, 4, 4, 10, 12, 13, 13, … ]
min: -9
max: 22
plus_count: 12
minus_count: 2
parni_count: 7
neparni_count: 7
double_count: 8
▶ Array(14) [ -9, -4, 1, 3, 4, 4, 10, 12, 13, 13, … ]
▶ Array(14) [ 22, 20, 19, 17, 13, 13, 12, 10, 4, 4, … ]
```

Результат виконання програми

Завдання 2.

- Розмістіть на сторінці 3 div-блоки з класами “**table**”. У кожному з них розмістіть табличку. У першому блоці - табличку розміром 3x3, другому - 4x4, третьому - 5x5.
- В CSS-файлі розмістіть стиль для класу “**selected**”, який передбачає жовтий фон.
- За допомогою **querySelectorAll** знайдіть **div**-блоки з класом “**table**”. Для кожного знайденого блоку у циклі знайдіть всі вкладені у нього теги **td**. Для кожного другого тега **td** додайте клас “**selected**”.

Лістинг style.css:

```
.selected {
    background-color: yellow;
}
```

Лістинг index.html:

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <meta charset="UTF-8">
```

```
    <title>Task 02</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
<div class="table">
    <table>
        <tr>
            <td>1</td>
            <td>2</td>
            <td>3</td>
        </tr>
        <tr>
            <td>4</td>
            <td>5</td>
            <td>6</td>
        </tr>
        <tr>
            <td>7</td>
            <td>8</td>
            <td>9</td>
        </tr>
    </table>
</div>

<div class="table">
    <table>
        <tr>
            <td>1</td>
            <td>2</td>
            <td>3</td>
            <td>4</td>
        </tr>
        <tr>
            <td>5</td>
            <td>6</td>
            <td>7</td>
            <td>8</td>
        </tr>
        <tr>
            <td>9</td>
            <td>10</td>
            <td>11</td>
            <td>12</td>
        </tr>
        <tr>
            <td>13</td>
            <td>14</td>
            <td>15</td>
            <td>16</td>
        </tr>
    </table>
</div>

<div class="table">
    <table>
        <tr>
            <td>1</td>
            <td>2</td>
            <td>3</td>
            <td>4</td>
            <td>5</td>
        </tr>
        <tr>
```

```
            <td>6</td>
            <td>7</td>
            <td>8</td>
            <td>9</td>
            <td>10</td>
        </tr>
        <tr>
            <td>11</td>
            <td>12</td>
            <td>13</td>
            <td>14</td>
            <td>15</td>
        </tr>
        <tr>
            <td>16</td>
            <td>17</td>
            <td>18</td>
            <td>19</td>
            <td>20</td>
        </tr>
        <tr>
            <td>21</td>
            <td>22</td>
            <td>23</td>
            <td>24</td>
            <td>25</td>
        </tr>
    </table>
</div>

<script>
    let tables = document.querySelectorAll('.table');

    tables.forEach(table => {
        let tds = table.querySelectorAll('td');
        tds.forEach((td, index) => {
            if ((index + 1) % 2 === 0) {
                console.log(td.innerHTML);
                td.classList.add('selected');
            }
        });
    });
</script>
</body>
</html>
```

```
1 2 3
4 5 6
7 8 9
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16
1  2  3  4  5
6  7  8  9  10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

Результат виконання програми

Завдання 3.

Створіть матрицю розміру N на M, яка містить цілі числа (наприклад, 7x5 та числами на власний розсуд). Знайдіть та виведіть у консоль:

- кількість додатних елементів;
- кількість рядків, які не містять жодного нульового елемента;
- кількість стовпців, які містять хоча б один нульовий елемент;
- номер рядка, в якому знаходиться найдовша серія однакових елементів;
- добуток елементів в тих рядках, які не містять від'ємних елементів;
- суму елементів в тих стовпцях, які не містять від'ємних елементів;
- суму елементів в тих стовпцях, які містять хоча б один від'ємний елемент;
- транспоновану матрицю;

ADVANCED TASKS:

* максимум серед сум елементів діагоналей, паралельних головній діагоналі матриці (*завдання на додаткові бали, приймається із обов'язковим усним поясненням та додатковими запитаннями від викладача*);

* мінімум серед сум модулів елементів діагоналей, паралельних побічній діагоналі матриці (*завдання на додаткові бали, приймається із обов'язковим усним поясненням та додатковими запитаннями від викладача*).

Лістинг програми:

```
<!DOCTYPE html>
<html lang="ua">
<head>
  <meta charset="UTF-8">
  <title>Task 03</title>
</head>
<body>
<script>
  let matrix = [
    [1, 2, 3, 0, -1],
    [4, 5, 6, 0, 7],
    [0, 1, 2, 3, 4],
    [-1, -2, 0, 1, 0],
    [5, 5, 5, 5, 5],
    [2, 3, 4, 0, 6],
    [7, 0, 8, 9, 10]
  ];

  let positiveCount = 0;
  let noZeroRowsCount = 0;
  let zeroColsCount = 0;
  let longestSeriesRow = -1;
  let longestSeriesLength = 0;
  let rowProducts = [];
  let colSumPositive = [];
  let colSumNegative = [];
  let transposeMatrix = [];

  matrix.forEach(row => {
    row.forEach(el => {
      if (el > 0) positiveCount++;
    });
  });

  matrix.forEach(row => {
    if (!row.includes(0)) {
      noZeroRowsCount++;
    }
  });

  let colsCount = matrix[0].length;
  for (let col = 0; col < colsCount; col++) {
    let hasZero = false;
    for (let row = 0; row < matrix.length; row++) {
      if (matrix[row][col] === 0) {
        hasZero = true;
        break;
      }
    }
    if (hasZero) zeroColsCount++;
  }

  matrix.forEach((row, rowIndex) => {
    let currentSeriesLength = 1;
    let maxSeriesLength = 1;
    for (let i = 1; i < row.length; i++) {
      if (row[i] === row[i - 1]) {
        currentSeriesLength++;
      } else {
        maxSeriesLength = Math.max(maxSeriesLength, currentSeriesLength);
        currentSeriesLength = 1;
      }
```

```
    }
    maxSeriesLength = Math.max(maxSeriesLength, currentSeriesLength);
    if (maxSeriesLength > longestSeriesLength) {
      longestSeriesLength = maxSeriesLength;
      longestSeriesRow = rowIndex + 1;
    }
  });

  matrix.forEach(row => {
    if (row.every(el => el >= 0)) {
      let product = row.reduce((acc, el) => acc * el, 1);
      rowProducts.push(product);
    }
  });

  for (let col = 0; col < colsCount; col++) {
    let sumPositive = 0;
    let sumNegative = 0;
    let hasNegative = false;

    for (let row = 0; row < matrix.length; row++) {
      if (matrix[row][col] >= 0) {
        sumPositive += matrix[row][col];
      } else {
        hasNegative = true;
        sumNegative += matrix[row][col];
      }
    }

    if (!hasNegative) {
      colSumPositive.push(sumPositive);
    } else {
      colSumNegative.push(sumNegative + sumPositive);
    }
  }

  for (let col = 0; col < colsCount; col++) {
    let newRow = [];
    for (let row = 0; row < matrix.length; row++) {
      newRow.push(matrix[row][col]);
    }
    transposeMatrix.push(newRow);
  }

  console.log(`Кількість додатних елементів: ${positiveCount}`);
  console.log(`Кількість рядків без нульових елементів: ${noZeroRowsCount}`);
  console.log(`Кількість стовпців з хоча б одним нульовим елементом: ${zero-
ColsCount}`);
  console.log(`Номер рядка з найдовшою серією однакових елементів: ${longest-
SeriesRow}`);
  console.log(`Добуток елементів в рядках без від'ємних елементів: ${rowProd-
ucts}`);
  console.log(`Сума елементів в стовпцях без від'ємних елементів: ${colSumPosi-
tive}`);
  console.log(`Сума елементів в стовпцях з від'ємними елементами: ${colSumNega-
tive}`);
  console.log('Транспонована матриця:');
  console.table(transposeMatrix);
</script>
</body>
</html>
```

```
Кількість додатних елементів: 25                                   index.html:107:11
Кількість рядків без нульових елементів: 1                         index.html:108:11
Кількість стовпців з хоча б одним нульовим елементом: 5            index.html:109:11
Номер рядка з найдовшою серією однакових елементів: 5              index.html:110:11
Добуток елементів в рядках без від'ємних елементів: 0,0,3125,0,0   index.html:111:11
Сума елементів в стовпцях без від'ємних елементів: 28,18           index.html:112:11
Сума елементів в стовпцях з від'ємними елементами: 18,14,31        index.html:113:11
Транспонована матриця:                                             index.html:114:11
console.table()                                                    index.html:115:11
```

(index)	0	1	2	3	4	5	6
0	1	4	0	-1	5	2	7
1	2	5	1	-2	5	3	0
2	3	6	2	0	5	4	8
3	0	0	3	1	5	0	9
4	-1	7	4	0	5	6	10

Результат виконання програми

Висновок: навчився створювати і обробляти одновимірні та багатовимірні масиви та ознайомився з операціями над ними.