МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Лабораторна робота №3

з дисципліни «Технології програмування»

Варіант 6

Виконав:

студент гр. ІР-21

Дирів Назар

Перевірив:

Бондаренко О.С.

Зараховано від

___.___._____

_____

(підпис викладача)

Київ-2025

ЗАВДАННЯ:

6	Написати функцію, яка перевіряє, чи число є простим. Якщо аргумент не число виводиться повідомлення про помилку.	Згенерувати масив із 15 випадкових чисел. Перевірити, чи є серед них від'ємні, і створити новий масив тільки з додатними числами.

```
// --- Функція перевірки простого числа ---
function isPrime(num) {
  if (typeof num !== "number" || !Number.isInteger(num)) {
    console.log(`Помилка: аргумент "${num}" не є цілим числом`);
    return false;
  }
  if (num < 2) return false; // Від'ємні числа, 0 і 1 не є простими
  for (let i = 2; i <= Math.sqrt(num); i++) {
    if (num % i === 0) return false;
  }
  return true;
}

// --- Генерація масиву з 15 випадкових чисел (можуть бути від'ємні) ---
function generateRandomArray(len = 15, min = -50, max = 100) {
  return Array.from({ length: len }, () =>
    Math.floor(Math.random() * (max - min + 1)) + min
  );
}

// --- Основна функція ---
function main() {
  const arr = generateRandomArray(15, -50, 100); // тепер можливі
від'ємні
  console.log("Згенерований масив:", arr.join(", "));

  const hasPrimes = arr.some(isPrime);
  if (!hasPrimes) {
    console.log("У масиві немає простих чисел.");
    return;
  }

  const primes = arr.filter(isPrime);
  console.log("Прості числа з масиву:", primes.join(", "));
}

main();
```

Контрольні запитання:

1. Що таке функція в JavaScript?
Функція — це блок коду, який можна викликати повторно для виконання певного завдання або обчислення значення.

2. Способи оголошення функцій і різниця:

- **Function Declaration:**

```
function foo() { return 1; }
```

Можна викликати **до оголошення** (hoisting).

- **Function Expression:**

```
const foo = function() { return 1; }
```

Можна викликати лише після визначення.

3. Стрілкова функція:

```
const foo = (x) => x * 2;
```

- Переваги: короткий синтаксис, не має власного `this`.
- Обмеження: не підходить для методів об’єкта та конструкторів.

4. Функції як об'єкти першого класу:
Можна присвоювати змінним, передавати як аргументи, повертати з інших функцій.

5. Параметри та аргументи:

- **Параметри** — змінні у визначенні функції.
- **Аргументи** — реальні значення при виклику.
- Перевіряти типи: `typeof arg === 'number'`, `Array.isArray(arg)` тощо.

6. Область видимості змінних:

- **Глобальна:** доступна скрізь.
- **Функціональна:** доступна всередині функції.
- **Блочна (let/const):** доступна тільки у блоці `{ }`.

7. Замикання (closure):
Функція пам’ятає змінні з зовнішньої області навіть після виходу зовнішньої функції.

```
function outer() {
```

```
  let x = 5;
  return function inner() { return x + 1; }
}
const fn = outer();
console.log(fn()); // 6
```

8. Рекурсія:
Функція викликає сама себе.

- Переваги: чистий код для складних структур (дерева, графи).
- Недоліки: можливе переповнення стеку, важче відлагоджувати.

9. Методи масивів:

- `forEach` — перебір без повернення нового масиву.
- `map` — створює новий масив із трансформованих елементів.
- `filter` — створює новий масив з елементів, що проходять умову.
- `reduce` — зводить масив до одного значення.

10. call, apply, bind:

- **call:** викликає функцію, передаючи `this` і аргументи окремо.

```
foo.call(obj, arg1, arg2);
```

- **apply:** викликає функцію, передаючи `this` і аргументи масивом.

```
foo.apply(obj, [arg1, arg2]);
```

- **bind:** повертає нову функцію з прив’язаним `this`.

```
const bound = foo.bind(obj);
bound(arg1);
```