# Лабораторна робота №3

**Тема:** Введення TypeScript.

**Мета:** Детальний огляд мови TypeScript, її переваг та основних концепцій для розробників JavaScript.

# Хід роботи

**Завдання:**

Переробіть завдання із практичної роботи №1 та №2 із JS на TS.

**Task 1:**

Код

const readline = require("readline/promises");

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

async function main(): Promise<void> {

    const day: number = Number(await rl.question("Enter day: "));

    const month: number = Number(await rl.question("Enter month: "));

    console.log("Time of year - " + getSeason(day, month));

    rl.close();

}

function getSeason(day: number, month: number){

    if (day < 1 || day > 31) {

        return "Wrong day";

    }

    if (month === 2 && day > 29) {

        return "May be Winter, but you entered the incorrect day!";

    }

    if (month < 1 || month > 12) {

        return "Wrong month";

    }

    return [12, 1, 2].includes(month)

        ? "winter"

        : month <= 5

        ? "spring"

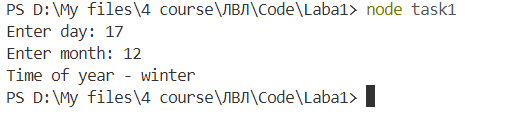
        : month <= 8

        ? "summer"

        : "autumn";

}

main();

Вивід:  


**Task 2:**

Код

import \* as readline from "readline/promises";

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

async function main(): Promise<void> {

    const count: number = Number(await rl.question("Enter count numbers: "));

    if (isNaN(count)) {

        console.error("Please enter a valid number!");

        rl.close();

        return;

    }

    printNumbers(count);

    rl.close();

}

function printNumbers(n: number): void {

    const numbers: number[] = [];

    for (let i = 0; i < n; i++) {

        if ((i % 2 === 0 || i % 3 === 0) && i % 6 !== 0) {

            numbers.push(i);

        }

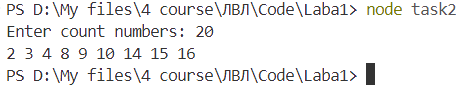
    }

    console.log(numbers.join(" "));

}

main();

Вивід



**Task 3**

Код

import \* as readline from "readline/promises";

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

async function main(): Promise<void> {

    const str: string = await rl.question("Enter string ");

    console.log("You have " + getCount(str) + " words what have 7 and more letters");

    rl.close();

}

function getCount(str: string){

      return str

    .split(/\s+/)

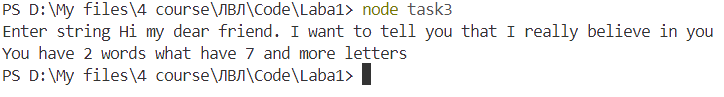
    .filter(word => word.length >= 7)

    .length;

}

main();

Вивід



**Task 4**

Код

const arrs: number[] = [];

for (let i = 0; i < 50; i++) {

    arrs[i] = Math.floor(Math.random() \* 100) + 1;

}

console.log("Початковий масив:");

console.log(arrs.join(" "));

const avgs: number = arrs.reduce((sum: number, num: number) => sum + num, 0) / arrs.length;

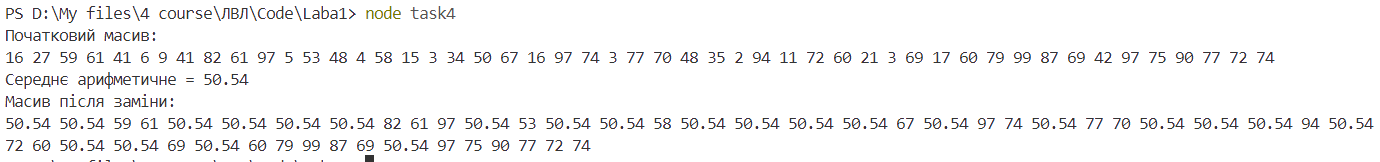
console.log("Середнє арифметичне =", avgs.toFixed(2));

const updatedArr: number[] = arrs.map((num: number) => num < avgs ? avgs : num);

console.log("Масив після заміни:");

console.log(updatedArr.join(" "));

Вивід



**Task 5**

Код

import \* as readline from "readline/promises";

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout

});

function transpose(matrix: number[][]): number[][] {

  return matrix[0].map((\_, colIndex) => matrix.map(row => row[colIndex]));

}

async function main(): Promise<void> {

  const rowsInput = await rl.question("Enter count rows: ");

  const colsInput = await rl.question("Enter count cols: ");

  const rows = parseInt(rowsInput);

  const cols = parseInt(colsInput);

  if (isNaN(rows) || isNaN(cols) || rows <= 0 || cols <= 0) {

    console.log("Please enter valid positive numbers for rows and columns.");

    rl.close();

    return;

  }

  let matrix: number[][] = [];

  for (let i = 0; i < rows; i++) {

    let line = await rl.question(`Enter elements row ${i + 1} (through a space): `);

    let numbers = line.trim().split(/\s+/).map(Number);

    if (numbers.length !== cols || numbers.some(isNaN)) {

      console.log(`Enter exactly ${cols} valid numbers separated by spaces.`);

      i--;

      continue;

    }

    matrix.push(numbers);

  }

  console.log("Початкова матриця:");

  console.table(matrix);

  let transposed = transpose(matrix);

  console.log("Транспонована матриця:");

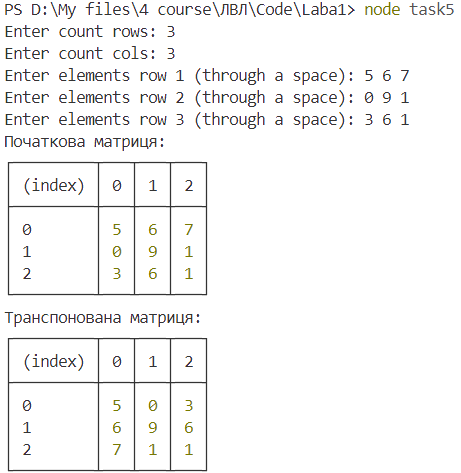
  console.table(transposed);

  rl.close();

}

main();

Вивід



**Task 6**

Код

import \* as readline from "readline/promises";

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout

});

async function main(): Promise<void> {

    let text: string = await rl.question("Enter text with <b>: ");

    text = text.replace(/<b>/g, "<strong>");

    text = text.replace(/<\/b>/g, "</strong>");

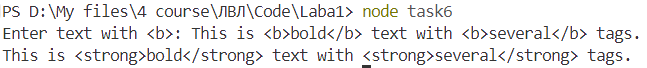
    console.log(text);

    rl.close();

}

main();

Вивід



**Task 7:**

Код

import \* as fs from 'fs';

interface Employee {

    lastName: string;

    firstName: string;

    middleName: string;

    department: number;

    lab: number;

    education: string;

    position: string;

    birthYear: number;

}

interface GroupedEmployees {

    [lab: number]: Employee[];

}

class EmployeeService {

    private employees: Employee[] = [];

    constructor() {

        this.loadEmployees();

    }

    private loadEmployees(): void {

        try {

            const data = fs.readFileSync('employees.json', 'utf8');

            const parsedData = JSON.parse(data);

            // Валідація даних

            if (Array.isArray(parsedData)) {

                this.employees = parsedData.map(emp => ({

                    lastName: String(emp.lastName || ''),

                    firstName: String(emp.firstName || ''),

                    middleName: String(emp.middleName || ''),

                    department: Number(emp.department) || 0,

                    lab: Number(emp.lab) || 0,

                    education: String(emp.education || ''),

                    position: String(emp.position || ''),

                    birthYear: Number(emp.birthYear) || 0

                }));

            } else {

                console.error("Invalid data format in employees.json");

                process.exit(1);

            }

        } catch (err) {

            console.error("Помилка читання файлу:", err);

            process.exit(1);

        }

    }

    public getEmployeesByDepartment(department: number): Employee[] {

        return this.employees.filter(emp => emp.department === department);

    }

    public groupByLab(employees: Employee[]): GroupedEmployees {

        const grouped: GroupedEmployees = {};

        employees.forEach(emp => {

            if (!grouped[emp.lab]) grouped[emp.lab] = [];

            grouped[emp.lab].push(emp);

        });

        return grouped;

    }

    public sortByLastName(employees: Employee[]): Employee[] {

        return [...employees].sort((a, b) => a.lastName.localeCompare(b.lastName));

    }

    public searchByName(name: string): Employee[] {

        const searchTerm = name.toLowerCase();

        return this.employees.filter(emp =>

            emp.firstName.toLowerCase().includes(searchTerm) ||

            emp.lastName.toLowerCase().includes(searchTerm) ||

            emp.middleName.toLowerCase().includes(searchTerm)

        );

    }

    public displayEmployees(employees: Employee[], title: string): void {

        console.log(`\n${title}:`);

        employees.forEach(emp => {

            console.log(`${emp.lastName} ${emp.firstName} ${emp.middleName}, ${emp.position}, лаб. ${emp.lab}`);

        });

    }

}

interface HRDepartment {

    person: string;

    rank: string;

    position: string;

    staffSchedule: string;

}

function main(): void {

    const employeeService = new EmployeeService();

    const departmentToFind = 1;

    const nameToSearch = "Марія";

    const depEmployees = employeeService.getEmployeesByDepartment(departmentToFind);

    employeeService.displayEmployees(depEmployees, `Працівники відділу ${departmentToFind}`);

    const groupedByLab = employeeService.groupByLab(depEmployees);

    console.log("\nЗгруповані по лабораторіях:");

    Object.keys(groupedByLab).forEach(lab => {

        console.log(`\nЛабораторія ${lab}:`);

        groupedByLab[Number(lab)].forEach(emp => {

            console.log(`  ${emp.lastName} ${emp.firstName}`);

        });

    });

    const sortedEmployees = employeeService.sortByLastName(depEmployees);

    employeeService.displayEmployees(sortedEmployees, "Відсортовані за прізвищем");

    const foundByName = employeeService.searchByName(nameToSearch);

    employeeService.displayEmployees(foundByName, `Результати пошуку за іменем '${nameToSearch}'`);

}

main();

Вивід:  


**Task 8:**

Код

import \* as fs from 'fs';

interface PersonData {

    firstName: string;

    lastName: string;

    age: number;

}

interface RankData {

    title: string;

}

interface PositionData {

    name: string;

}

interface StaffMember {

    person: Person;

    rank: Rank;

    position: Position;

}

interface StaffDataFromFile {

    firstName: string;

    lastName: string;

    age: number;

    rank: string;

    position: string;

}

class Person {

    firstName: string;

    lastName: string;

    age: number;

    constructor(firstName: string, lastName: string, age: number) {

        this.firstName = firstName;

        this.lastName = lastName;

        this.age = age;

    }

}

class Rank {

    title: string;

    constructor(title: string) {

        this.title = title;

    }

}

class Position {

    name: string;

    constructor(name: string) {

        this.name = name;

    }

}

class StaffingTable {

    private staff: StaffMember[] = [];

    addPerson(person: Person, rank: Rank, position: Position): void {

        this.staff.push({ person, rank, position });

    }

    loadFromFile(filename: string): void {

        try {

            const data = fs.readFileSync(filename, 'utf8');

            const staffData: StaffDataFromFile[] = JSON.parse(data);

            staffData.forEach(item => {

                const person = new Person(item.firstName, item.lastName, item.age);

                const rank = new Rank(item.rank);

                const position = new Position(item.position);

                this.addPerson(person, rank, position);

            });

            console.log(`Дані успішно завантажено з файлу ${filename}`);

        } catch (err) {

            console.error("Помилка читання файлу:", err);

        }

    }

    sortByRank(): StaffMember[] {

        return [...this.staff].sort((a, b) => a.rank.title.localeCompare(b.rank.title));

    }

    sortByPosition(): StaffMember[] {

        return [...this.staff].sort((a, b) => a.position.name.localeCompare(b.position.name));

    }

    searchByName(name: string): StaffMember[] {

        const searchTerm = name.toLowerCase();

        return this.staff.filter(s =>

            s.person.firstName.toLowerCase().includes(searchTerm) ||

            s.person.lastName.toLowerCase().includes(searchTerm)

        );

    }

    showAll(): void {

        console.log("\nВесь список співробітників:");

        this.staff.forEach(s => {

            console.log(`${s.person.firstName} ${s.person.lastName}, Вік: ${s.person.age}, Звання: ${s.rank.title}, Посада: ${s.position.name}`);

        });

    }

}

function main(): void {

    const hr = new StaffingTable();

    hr.loadFromFile('staff.json');

    hr.showAll();

    console.log("\nСортування по званню:");

    hr.sortByRank().forEach(s => {

        console.log(`${s.rank.title}: ${s.person.firstName} ${s.person.lastName}`);

    });

    console.log("\nПошук по імені 'Іван':");

    const found = hr.searchByName("Іван");

    found.forEach(f => {

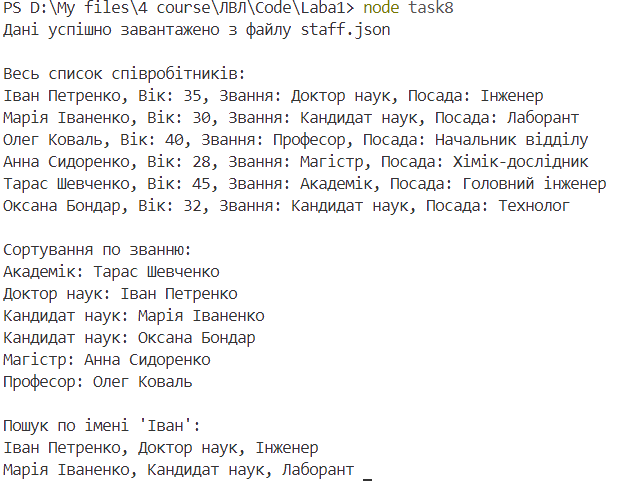
        console.log(`${f.person.firstName} ${f.person.lastName}, ${f.rank.title}, ${f.position.name}`);

    });

}

main();

Вивід



**Висновок:** я освоїла мову TypeScript, її переваги та основні концепції для розробників JavaScript.