**Державний вищий навчальний заклад  
Ужгородський національний університет  
Факультет інформаційних технологій**

ПРАКТИЧНА РОБОТА № 4  
**Тема:** Алгоритми з розгалуженням. Алгоритми з циклами.

Виконав студент  
І курсу спеціальності «Інженерія програмного забезпечення»

Романюк Артем

**Ужгород-2025**

**Мета:** навчитися складати й програмувати алгоритми з послідовною перевіркою умов; алгоритми з розгалуженням, використовуючи вкладені умовні оператори та оператори вибору; алгоритми з циклами.

**Питання для самоконтролю**

1. Охарактеризуйте алгоритми з розгалуженнями. Чим вони відрізняються від лінійних алгоритмів? У яких випадках використовують алгоритми з розгалуженнями?
2. Які оператори використовують для програмування розгалужень? Охарактеризуйте різні форми умовного оператора і їхнє використання. Охарактеризуйте оператор вибору. Коли його доцільно використовувати?
3. На що вказує цикломатична складність алгоритму? Як для алгоритму побудувати граф керування? Як цикломатична складність алгоритму пов’язана з цикломатичним числом графа керування?
4. Що таке цикл? Для чого використовують цикли? Що називають тілом циклу? Чи може бути тіло циклу порожнім? Які алгоритми називають циклічними?
5. Яких типів є оператори циклу і чи є між ними відмінності при виконанні? У яких випадках надають перевагу циклу з параметром, циклу з передумовою чи циклу з післяумовою? Чи можуть бути вкладеними один в оден цикли різних типів? Які вкладені цикли називають залежними, а які незалежними?
6. Як визначити часову складність циклів з параметром, з передумовою і післяумовою? Як оптимізувати виконання циклічного алгоритму за часом?

Завдання до роботи:

1. Написати код для виконання алгоритму відповідно до варіанту завдання 1 та 2.
2. Оформити звіт та завантажити його в системі електронного навчання ДВНЗ «УжНУ» в установлений термін.
3. Підготувати відповіді на контрольні питання.

**Хід роботи:**

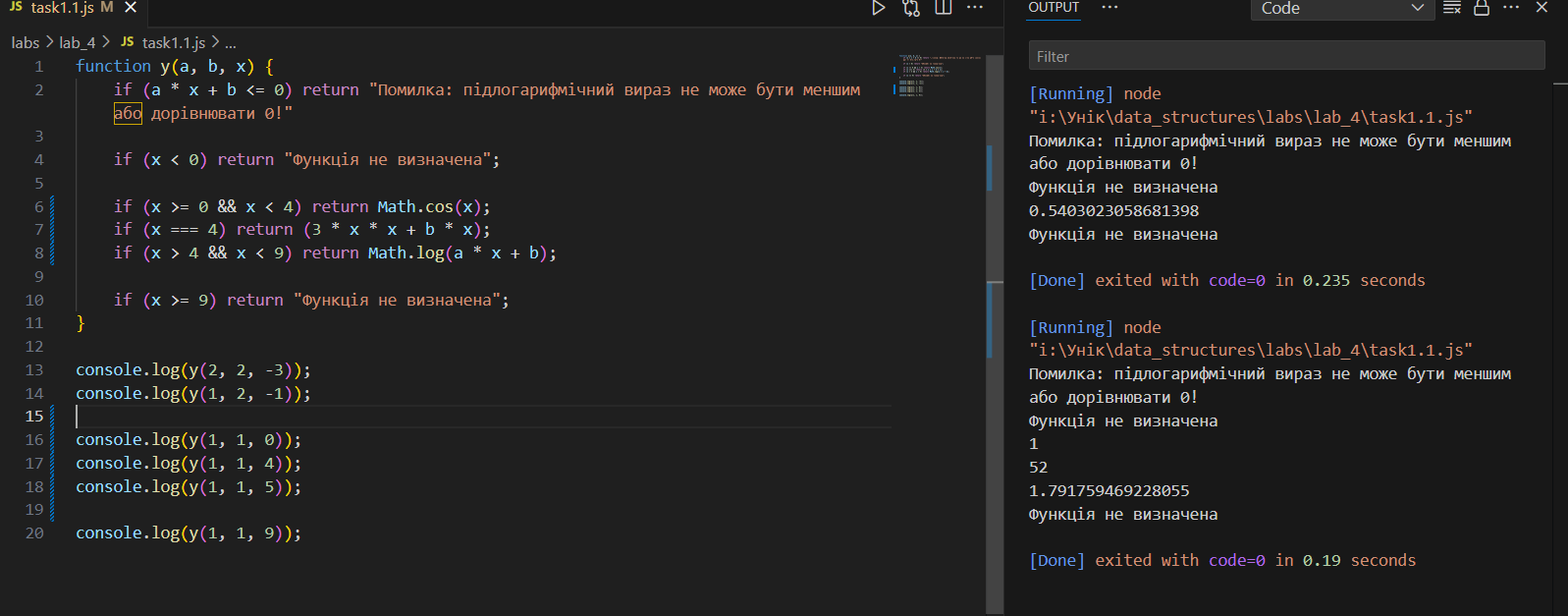
**Варіант 27 (12)**

**Завдання 1.1.**

Розробити алгоритм з послідовною перевіркою умов (неповна форма умовного оператора) для обчислення значень заданої логічною залежністю функції f(x) при довільних значеннях параметрів а і b і незалежної змінної х. Видачу повідомлень про помилки здійснювати зразу ж при їхньому виявленні, а видачу результату обчислення значення функції f(x) — у кінці алгоритму. В алгоритмі передбачити перевірку, чи немає ділення на нуль, чи підкореневий вираз невід’ємний, чи аргумент функції логарифма набуває додатних значень тощо. Вважається, що функція визначена на заданому проміжку, а поза ним не визначена.

Реалізація в коді:

| function y(a, b, x) {  if (a \* x + b <= 0) return "Помилка: підлогарифмічний вираз не може бути меншим або дорівнювати 0!"  if (x < 0) return "Функція не визначена";  if (x >= 0 && x < 4) return Math.cos(x);  if (x === 4) return (3 \* x \* x + b \* x);  if (x > 4 && x < 9) return Math.log(a \* x + b);  if (x >= 9) return "Функція не визначена";  }  console.log(y(2, 2, -3));  console.log(y(1, 2, -1));  console.log(y(1, 1, 0));  console.log(y(1, 1, 4));  console.log(y(1, 1, 5));  console.log(y(1, 1, 9)); |
| --- |

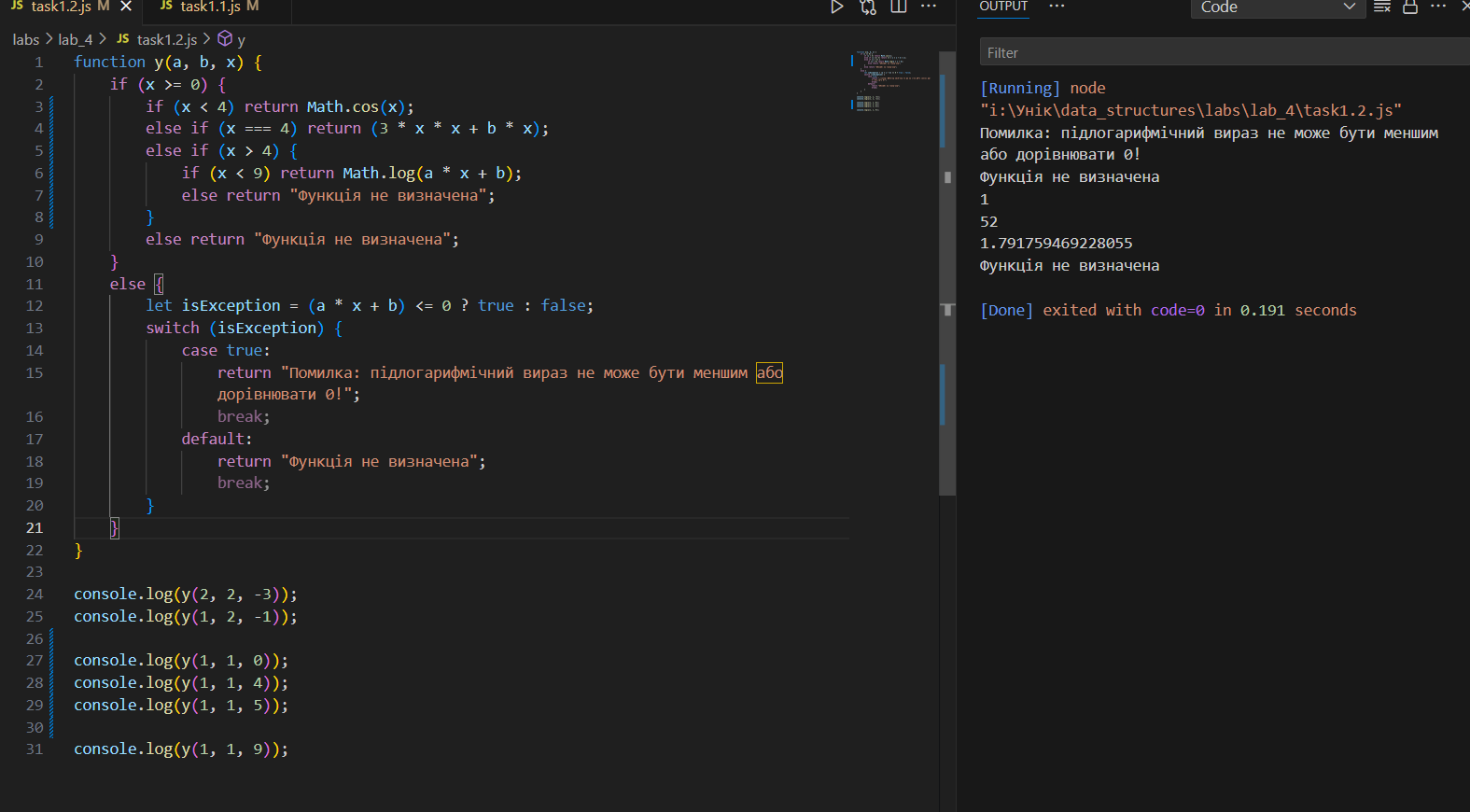
****

**Завдання 1.2.** Розробити алгоритм з вкладеною перевіркою умов (порівняння в умовах не повторювати) для обчислення значень заданої логічною залежністю функції f(x) при довільних значеннях параметрів а і b і незалежної змінної х.

**(\*)** Вивід повідомлень про помилки зробити вкінці алгоритму за допомогою оператора вибору.

Реалізація в коді:

| function y(a, b, x) {  if (x >= 0) {  if (x < 4) return Math.cos(x);  else if (x === 4) return (3 \* x \* x + b \* x);  else if (x > 4) {  if (x < 9) return Math.log(a \* x + b);  else return "Функція не визначена";  }  else return "Функція не визначена";  }  else {  let isException = (a \* x + b) <= 0 ? true : false;  switch (isException) {  case true:  return "Помилка: підлогарифмічний вираз не може бути меншим або дорівнювати 0!";  break;  default:  return "Функція не визначена";  break;  }  }  }  console.log(y(2, 2, -3));  console.log(y(1, 2, -1));  console.log(y(1, 1, 0));  console.log(y(1, 1, 4));  console.log(y(1, 1, 5));  console.log(y(1, 1, 9)); |
| --- |

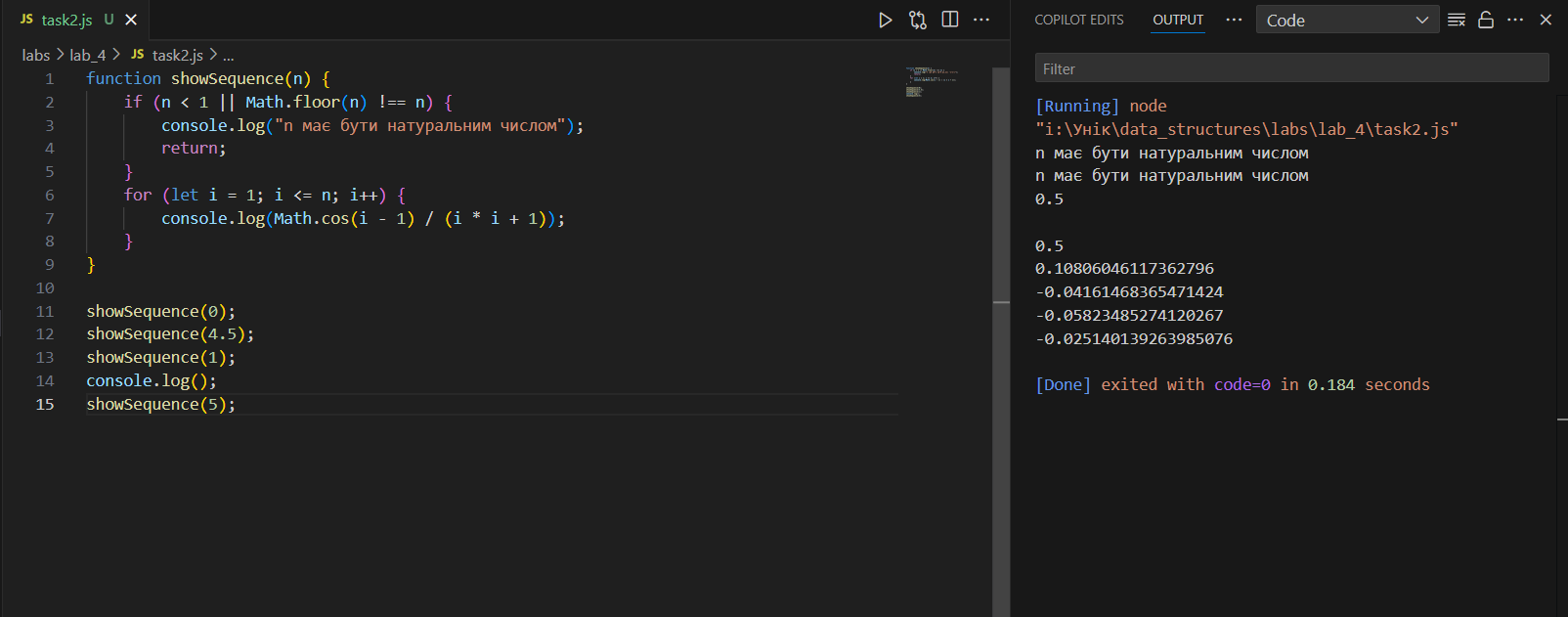


**Завдання 2.**

Розробити алгоритм розв’язування задачі: перші *m* (*m* ≥ 1) членів послідовності {*xn*} (*n* = 1, 2, 3,…) вивести на екран в стовпчик. В алгоритмі передбачити перевірку правильності введення даних.

Реалізація в коді:

| function showSequence(n) {  if (n < 1 || Math.floor(n) !== n) {  console.log("n має бути натуральним числом");  return;  }  for (let i = 1; i <= n; i++) {  console.log(Math.cos(i - 1) / (i \* i + 1));  }  }  showSequence(0);  showSequence(4.5);  showSequence(1);  console.log();  showSequence(5); |
| --- |



**Висновок:**

У даній роботі було розглянуто алгоритми з розгалуженням і алгоритми з циклами, які є невід'ємною частиною програмування та алгоритмізації.

Алгоритми з розгалуженням дозволяють реалізовувати вибір між кількома варіантами дій, що забезпечує гнучкість програм. Основними конструкціями таких алгоритмів є умовні оператори (if, else, switch), які використовуються для прийняття рішень на основі заданих умов.

Алгоритми з циклами забезпечують можливість багаторазового повторення певних дій, що є необхідним для обробки великих обсягів даних або виконання однотипних операцій. Основними видами циклів є for, while та do-while, кожен з яких має свої особливості застосування.

Завдяки використанню розгалужень і циклів можна створювати складні та ефективні алгоритми, що дозволяють оптимізувати обчислення, автоматизувати повторювані процеси та реалізовувати логіку програм. Розуміння цих конструкцій є необхідним для подальшого вивчення алгоритмів і програмування загалом.