**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»

Боднарчук Роман Романович

**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.

**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 завдання 1 (Варіант 2)**

**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.

**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.

function calculateFunction(x, a, b) {

  let result;

  let errorCode = 0;

  if (x >= 0 && x <= 8) {

      if (x < 5) {

          result = Math.cos(x);

      } else if (x === 5) {

          let sqrtValue = a \* x - b;

          if (sqrtValue < 0) {

              errorCode = 1;

          } else if (Math.sqrt(sqrtValue) === 0) {

              errorCode = 2;

          } else {

              result = 1 / Math.sqrt(sqrtValue);

          }

      } else {

          result = b \* x \*\* 2 + 3 \* x - 2;

      }

  } else {

      errorCode = 3;

  }

  switch (errorCode) {

      case 1:

          return "Помилка: підкореневий вираз від’ємний";

      case 2:

          return "Помилка: ділення на нуль";

      case 3:

          return "Помилка: x поза допустимим діапазоном";

      default:

          return `f(${x}) = ${result}`;

  }

}

// Приклад

let x = 5;

let a = 2;

let b = 3;

console.log(calculateFunction(x, a, b));

**Варіант завдання 2 (Варіант 2)**

**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.

function calculateSequence(m) {

  if (m < 1 || !Number.isInteger(m)) {

      console.log("m має бути цілим числом і не менше 1");

      return;

  }

  for (let n = 1; n <= m; n++) {

      let xn = (Math.sin(2 \* n + 3) - 2) / (n + 1);

      console.log(`x\_${n} = ${xn}`);

  }

}

// Приклад

let m = 5;

calculateSequence(m);

**Питання для самоконтролю**

1. Охарактеризуйте алгоритми з розгалуженнями. Чим вони відрізняються від лінійних алгоритмів? У яких випадках використовують алгоритми з розгалуженнями?

**Відповідь:** Алгоритми з розгалуженнями відрізняються від лінійних алгоритмів тим, що вони можуть змінювати свій хід виконання в залежності від певних умов. У лінійних алгоритмах виконання відбувається послідовно, без розгалужень.

Алгоритми з розгалуженнями використовуються тоді, коли потрібно виконати певний блок коду або дію, якщо виконуються певні умови (наприклад, якщо число більше за певне значення), або виконати інший блок коду, якщо умови не виконуються. Вони дозволяють програмі реагувати на різні сценарії та змінювати свою поведінку відповідно до умов.

1. Які оператори використовують для програмування розгалужень? Охарактеризуйте різні форми умовного оператора і їхнє використання. Охарактеризуйте оператор вибору. Коли його доцільно використовувати?

**Відповідь:** Для програмування розгалужень використовуються умовні оператори. Вони дозволяють виконувати різний код в залежності від умови.

Умовний оператор може мати кілька форм:

**- if** - виконує певний блок коду, якщо умова істинна.

**- if-else** - виконує один блок коду, якщо умова істинна, інакше інший блок коду.

**- if-else if-else** - дозволяє перевірити декілька умов по черзі і виконати відповідний блок коду для першої істинної умови.

**- switch** - оператор вибору, який дозволяє вибирати один з багатьох блоків коду для виконання, в залежності від значення виразу.

Оператор вибору switch доцільно використовувати, коли потрібно вибрати один з багатьох варіантів в залежності від значення виразу. Це дозволяє зробити код більш зрозумілим та компактним, особливо якщо є багато варіантів у виборі.

1. На що вказує цикломатична складність алгоритму? Як для алгоритму побудувати граф керування? Як цикломатична складність алгоритму пов’язана з цикломатичним числом графа керування?

**Відповідь:** Цикломатична складність алгоритму вказує на його складність, вимірюючи кількість незалежних шляхів у коді. Для побудови графа керування алгоритму потрібно представити кожен блок коду як вузол, а умови переходу між блоками як ребра. Цикломатична складність пов'язана з цикломатичним числом графа керування, яке можна обчислити за допомогою формули V(G) = E - N + 2P, де E - кількість ребер, N - кількість вузлів, P - кількість компонентів зв’язності графа. Цикломатична складність алгоритму дорівнює цикломатичному числу графа керування плюс одиниця.

1. Що таке цикл? Для чого використовують цикли? Що називають тілом циклу? Чи може бути тіло циклу порожнім? Які алгоритми називають циклічними?

**Відповідь:** Цикл - це структура програмування, яка дозволяє повторювати виконання певного блоку коду декілька разів, на підставі заданої умови.

Цикли використовуються для автоматизації повторюваних завдань та обробки даних у списку, масиві чи іншій структурі даних.

Тіло циклу - це блок коду, який виконується кожен раз під час ітерації циклу. Так, тіло циклу може бути порожнім, коли не потрібно виконувати жодної дії під час кожної ітерації.

Алгоритми, які використовують цикли для повторення дій, називають циклічними алгоритмами

1. Яких типів є оператори циклу і чи є між ними відмінності при вико- нанні? 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?

**Відповідь:** Оператори циклу можуть бути наступних типів:

**- Цикл з передумовою (while)** - виконується, поки умова істинна.

**- Цикл з післяумовою (do-while)** - виконується принаймні один раз, а потім поки умова істинна.

**- Цикл з параметром (for)** - виконується заздалегідь відому кількість разів.

Відмінності між ними полягають у порядку виконання та моменті перевірки умови. Наприклад, у циклі while умова перевіряється перед кожною ітерацією, у циклі do-while - після кожної ітерації, а у циклі for - на початку кожної ітерації.

У випадках, коли кількість ітерацій відома наперед, зазвичай надають перевагу циклу з параметром (for). Цикли з передумовою (while) та післяумовою (do-while) використовуються тоді, коли умова повторення не пов'язана з кількістю ітерацій.

Цикли можуть бути вкладеними один в одного різних типів. Вкладені цикли називають залежними, коли зміна у зовнішньому циклі впливає на внутрішній цикл, і незалежними, коли це не відбувається.

1. Як визначити часову складність циклів з параметром, з передумовою і післяумовою? Як оптимізувати виконання циклічного алгоритму за часом?

**Відповідь:** Часова складність циклів з параметром, з передумовою і післяумовою визначається кількістю ітерацій циклу та часом виконання однієї ітерації.

**- Для циклу з параметром (for)** часова складність визначається кількістю ітерацій, тобто O(n), де n - кількість ітерацій.

**- Для циклу з передумовою (while) або післяумовою (do-while)** часова складність також залежить від кількості ітерацій, але вона може бути різною в залежності від умови.

Оптимізація виконання циклічного алгоритму за часом включає в себе наступні підходи:

**1)** Використання циклу з параметром, коли можливо, оскільки він дозволяє явно вказати кількість ітерацій і зменшити витрати на перевірку умови.

**2)** Уникання зайвих обчислень або операцій у тілі циклу.

**3)** Мінімізація вкладених циклів та оптимізація алгоритму так, щоб він працював швидше у середньому випадку.

**4)** Використання ефективних структур даних та алгоритмів для обробки даних у циклі.

**Висновки:** У ході виконання Практичної роботи №4, я написав Алгоритм із розгалуженням та циклами який розв’язує математичні рівняння. Зробив алгоритм з вкладеною перевіркою умов для обчислення значень заданої логічною залежністю функції f(x) при довільних значеннях параметрів а і b і незалежної змінної х. Також розробив алгоритм розв’язування задачі: перші *m* (*m* ≥ 1) членів послідовності {*xn*} (*n* = 1, 2, 3,…) вивів на екран в стовпчик. В алгоритмі передбачив перевірку правильності введення даних. Зробив вивід повідомлень про помилки вкінці алгоритму за допомогою оператора вибору. Тим самим набув практичних навичок із виконанням алгоритмів та зробив висновок, що ці конструкції дуже важливі для створення ефективних та логічних алгоритмів у програмуванні.

**Висновок:**

В ході лабораторної роботи я розробив алгоритми для обчислення значень функцій та послідовностей із використанням умовних операторів та циклів.

Спочатку я реалізував алгоритм з послідовною перевіркою умов для функції f(x), враховуючи можливі помилки, такі як ділення на нуль або підкореневий вираз, що не має дійсних значень. Далі я вдосконалив алгоритм, використовуючи вкладену перевірку умов та оператор вибору для виводу повідомлень про помилки.

Також я розробив алгоритм для обчислення перших m членів послідовності {xn}, реалізувавши перевірку введених даних та організувавши вивід значень у стовпчик.

У результаті виконання роботи я закріпив навички роботи з умовними операторами, вкладеними перевірками, циклами та перевіркою вхідних даних у JavaScript.