*Додаток 1*

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни «Алгоритми та структури даних-1. Основи алгоритмізації»

«Дослідження ітераційних циклічних алгоритмів»

Варіант 34

Виконав студент ІП-13 Шиманська Ганна Артурівна
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
( прізвище, ім'я, по батькові)

Київ 2021

## Лабораторна робота 3

## Дослідження ітераційних циклічних алгоритмів

**Мета** – дослідити подання операторів повторення дій та набути практичних навичок їх використання під час складання циклічних програмних специфікацій.

### Варіант 34

34. З точністю $\varepsilon = 10^{-8}$ обчислити значення функції $\frac{e^x - e^{-x}}{2}$ за формулою $S = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \frac{x^9}{9!} + \ldots$, використавши рекурентну формулу для обчислення члена ряду.

- **Постановка задачі**

  Залежно від числа x обчислити значення функції із заданою точністю.

- **Побудова математичної моделі**

Складемо таблицю змінних

| ***Змінна*** | ***Тип*** | ***Ім'я*** | ***Призначення*** |
|---|---|---|---|
| Аргумент функції | Дійсний | x | Вхідні дані |
| Номер ітерації | Цілочисельний, > 0 | i | Проміжні дані |
| Точність | Дійсний | e | Проміжні дані |
| i-тий член рекурентної формули | Дійсний | expression | Проміжні дані |

| | | | |
|---|---|---|---|
| Чисельник виразу | Дійсний | differenceNumerator | Проміжні дані |
| Знаменник виразу | Дійсний | differenceDenominator | Проміжні дані |
| Значення функції | Дійсний | sum | Вихідні дані |

Точність e задамо за допомогою функції pow, яка повертає число, піднесене до заданого степеня. Умову на задану точність реалізуємо, використовуючи функцію fabs, яка повертає модуль числа через тип даних double. Для піднесення до степеня чисельника виразу повторно застосуємо pow, замінивши один з параметрів функції на змінну x, ініціалізовану користувачем за допомогою вхідного потоку cin.

**Розв’язання**

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми**.**

*Крок 1.* Визначимо основні дії.

*Крок 2.* Деталізуємо знаходження чисельника виразу differenceNumerator.

*Крок 3.* Ініціалізуємо змінну expression через вхідне дане x та змінну i.

*Крок 4.* Деталізуємо дію знаходження sum.

*Крок 5.* Деталізуємо знаходження знаменника виразу differenceDenominator.

*Крок 6.* Деталізуємо дію знаходження expression.

- **Псевдокод алгоритму**

*Крок 1.*

**початок**

Введення x

Обчислення differenceNumerator

Ініціалізація expression та i

Обчислення sum

Обчислення differenceDenominator

Обчислення expression

Виведення sum

**кінець**

*Крок 2.*

**початок**

Введення x

differenceNumerator = pow(x, 2)

Ініціалізація expression та i

Обчислення sum

Обчислення differenceDenominator

Обчислення expression

Виведення sum

**кінець**

*Крок 3.*

**початок**

Введення x

differenceNumerator = pow(x, 2)

expression = x

i = 0

Обчислення sum

Обчислення differenceDenominator

Обчислення expression

Виведення sum

**кінець**

*Крок 4.*

**початок**

Введення x

differenceNumerator = pow(x, 2)

expression = x

i = 0

**поки** fabs(expression) > e

**повторити**

sum += expression

Обчислення differenceDenominator

Обчислення expression

**все повторити**

Виведення sum

**кінець**

*Крок 5.*

**початок**

Введення x

differenceNumerator = pow(x, 2)

expression = x

i = 0

**поки** fabs(expression) > e

**повторити**

sum += expression

i++

differenceDenominator  = (i * 2) * ((i * 2) + 1)

Обчислення expression

**все повторити**

Виведення sum

**кінець**

*Крок 6.*

**початок**

Введення x

differenceNumerator = pow(x, 2);

expression = x

i = 0;

**поки** fabs(expression) > e

**повторити**

sum += expression;

i++

differenceDenominator  = (i * 2) * ((i * 2) + 1)

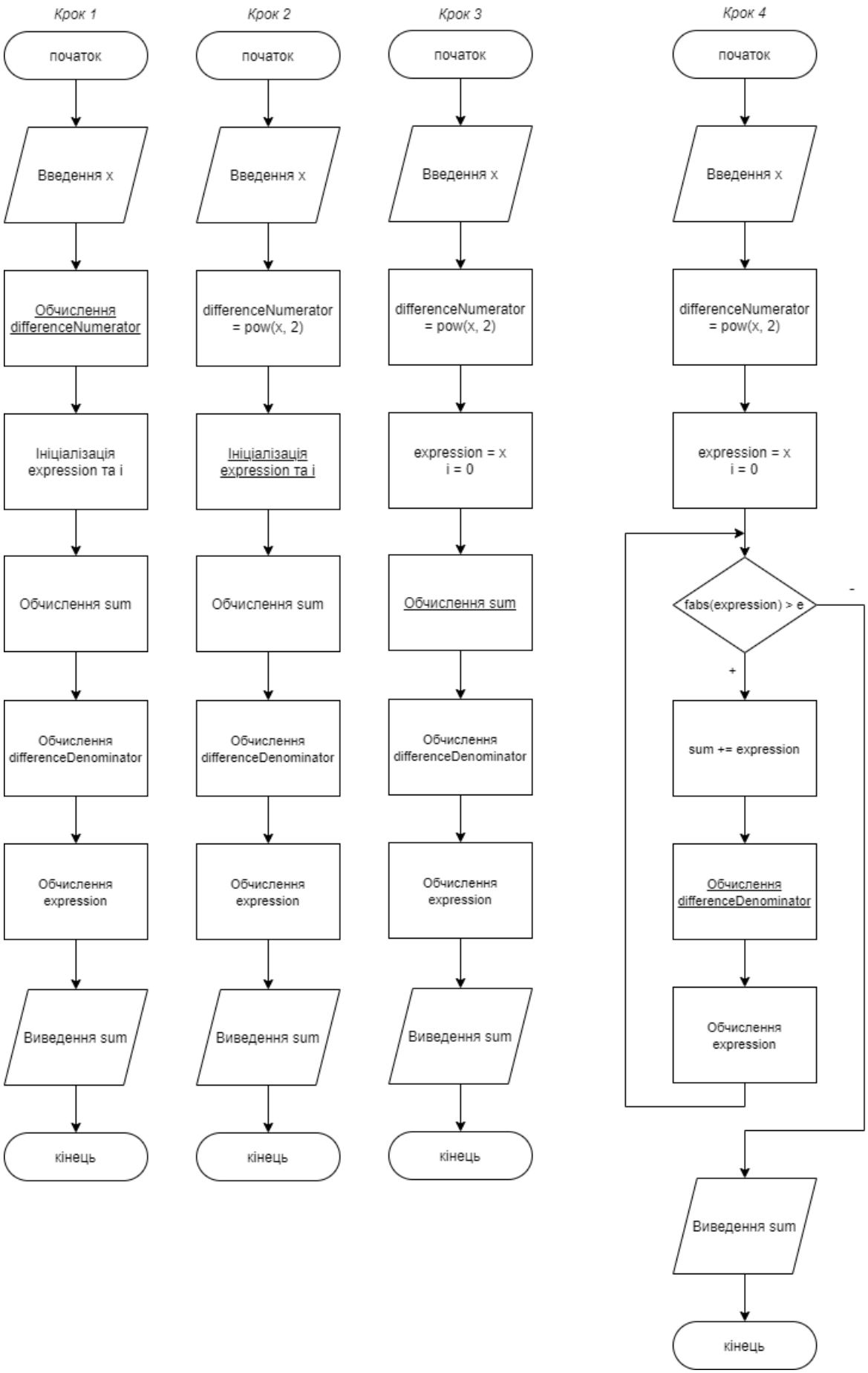
expression *= differenceNumerator/differenceDenominator

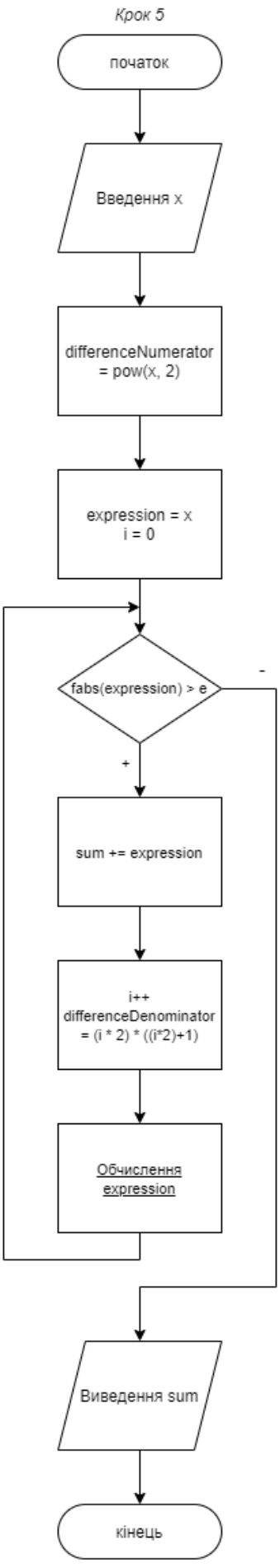
**все повторити**

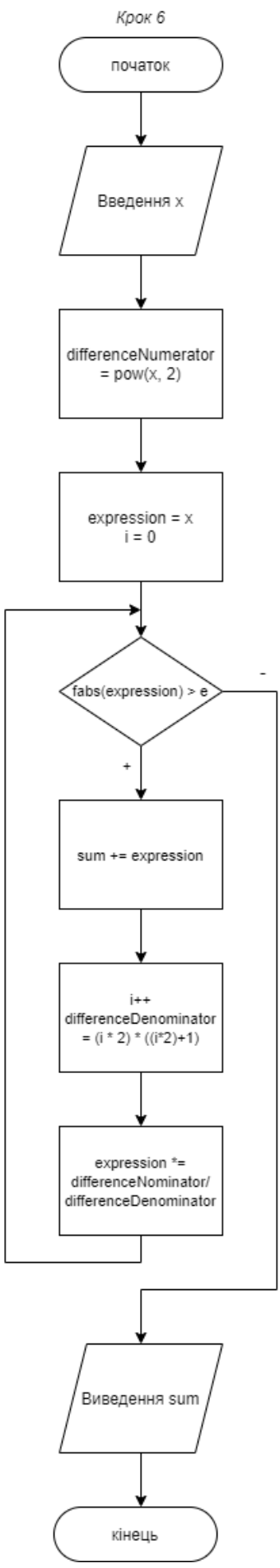
Виведення sum

**кінець**

- **Блок-схема**


Крок 1
початок
Введення x
Обчислення differenceNumerator
Ініціалізація expression та i
Обчислення sum
Обчислення differenceDenominator
Обчислення expression
Виведення sum
кінець
Крок 2
початок
Введення x
differenceNumerator = pow(x, 2)
Ініціалізація expression та i
Обчислення sum
Обчислення differenceDenominator
Обчислення expression
Виведення sum
кінець
Крок 3
початок
Введення x
differenceNumerator = pow(x, 2)
expression = x
i = 0
Обчислення sum
Обчислення differenceDenominator
Обчислення expression
Виведення sum
кінець
Крок 4
початок
Введення x
differenceNumerator = pow(x, 2)
expression = x
i = 0
fabs(expression) > e
-
+
sum += expression
Обчислення differenceDenominator
Обчислення expression
Виведення sum
кінець


Крок 5
початок
Введення x
differenceNumerator = pow(x, 2)
expression = x
i = 0
fabs(expression) > e
-
+
sum += expression
i++
differenceDenominator = (i * 2) * ((i*2)+1)
Обчислення expression
Виведення sum
кінець


Крок 6
початок
Введення x
differenceNumerator = pow(x, 2)
expression = x
i = 0
fabs(expression) > e
-
+
sum += expression
i++
differenceDenominator = (i * 2) * ((i*2)+1)
expression *= differenceNominator/ differenceDenominator
Виведення sum
кінець

- **Випробування алгоритму**

| ***Номер ітерації i*** | ***Дія*** |
|---|---|
| | початок |
| | Введення x = 5 |
| 1 | sum = 5.00000000 |
| 2 | sum = 25.83333333 |
| 3 | sum = 51.87500000 |
| 4 | sum = 67.37599206 |
| 5 | sum = 72.75828097 |
| 6 | sum = 73.98152845 |
| 7 | sum = 74.17756170 |
| 8 | sum = 74.20089900 |
| 9 | sum = 74.20304397 |
| 10 | sum = 74.20320076 |
| 11 | sum = 74.20321010 |
| 12 | sum = 74.20321056 |
| 13 | sum = 74.20321058 |
| | Виведення sum = 74.20321058 |
| | кінець |

- **Висновки:**

Розв’язавши цю задачу я навчилася працювати з операторами повторення дій та склала блоксхему циклічної програмної специфікації. За допомогою ітерацій я змогла визначити значення функції та прослідкувати за його зміною на кожному з кроків, поки наступний елемент ряду задовольняв задану умову.