

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів

Лабораторна робота № 5

з дисципліни «Алгоритмізація та програмування»
на тему «Реалізація циклічних алгоритмів мовою C ++»

XAI.301.319.05 ЛР

Виконав студент гр. _____319_____

_____Валерій ЄВЧУК_____

(підпис, дата)

(П.І.Б.)

Перевірив

_____к.т.н., доц. Євгеній ПЯВКА

(підпис, дата)

(П.І.Б.)

МЕТА РОБОТИ

Вивчити теоретичний матеріал із синтаксису мовою C ++ і поданням у вигляді UML діаграм циклічних алгоритмів і реалізувати алгоритми з використанням інструкцій циклу з передумовою, циклу з післяумовою і параметризованого циклу мовою C ++ в середовищі Visual Studio.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Дано дійсні числа (x_i, y_i) , $i = 1, 2, \dots, n$, – координати точок на площині. Визначити кількість точок, що потрапляють в фігуру заданого кольору (або групу фігур). Варіанти фігур представлено в табл.1. Дано n точок (x_i, y_i) . Визначити кількість точок, що належать заданій фігурі (області) при параметрі a .

Обмеження (квадрат): $x \in [-a/2; a/2]$, $y \in [-a; 0]$.

Коло (вписане в квадрат):

$$O(0; -a/2), r = a/2,$$

$$x^2 + (y + a/2)^2 = (a/2)^2.$$

Пряма (діагональ під 45°):

$$y = x - a/2$$

Завдання 2. Дано дійсне число x і натуральне число n . Необхідно: а) Обчислити значення виразу при заданих x і n для виразу з табл.2. б) Вивести: для парних варіантів – значення кожного третього елемента, для непарних – значення кожного четвертого елемента.

Дано дійсне x та натуральне n . Обчислити: $S = -1 + \sum_{k=0..n} [(-1)^k \cdot (-\pi/2 + \sqrt[4]{x})^{(2k)}] / (2k)!$.

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: $|u_n| < \epsilon$ або $|u_n| > g$, де ϵ – мала величина для переривання циклу обчислення суми збіжного ряду ($\epsilon = 10^{-5} \dots 10^{-20}$); g – величина для переривання циклу обчислення суми розбіжного ряду ($g = 10^2 \dots 10^5$). Варіанти представлено в табл.3.

Завдання 4. Організувати меню в командному вікні для багаторазового виконання завдань. *Додати функцію/функції, що вводять з консолі та повертають коректне значення цілого/дійсного типу у відповідності з обмеженнями вхідних даних кожного завдання.

Дослідити ряд на збіжність і обчислювати суму до виконання умови зупинки.

$$u_n = (n! - 3^n) / n^n, \sum_{n=1.. \infty} u_n.$$

Умова зупинки для збіжного: $|u_n| < \epsilon$, де $\epsilon \in [10^{-20}; 10^{-5}]$.

Завдання 5. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів:

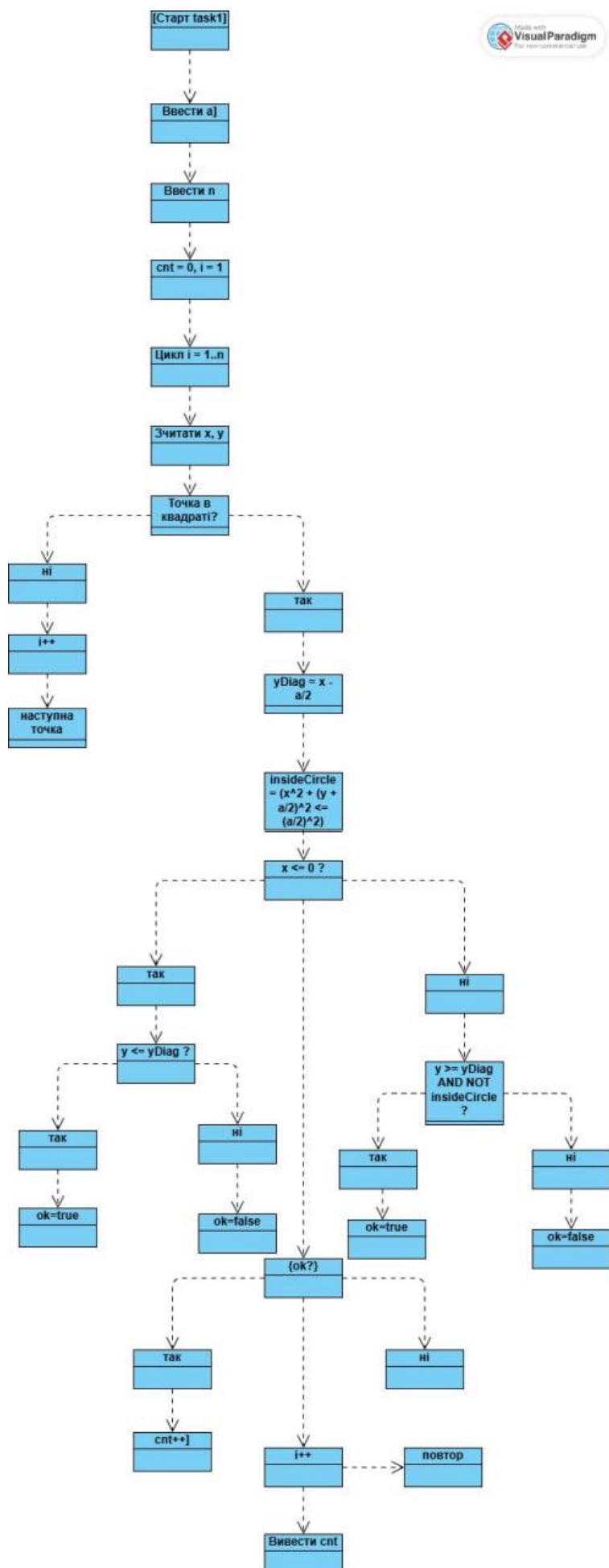
ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі (розділ і номер задачі): ЛР5, Завдання 1 (табл.1 – 10, Figure=10).

Вхідні дані (ім'я, опис, тип, обмеження): a — параметр фігури (сторона квадрата), long double, $a > 0$; n — кількість точок, int, $1 \leq n \leq 100000$; x, y —

координати точки, long double (вводяться для кожної точки), для перевірки належності квадрату використовується умова $x \in [-a/2; a/2]$, $y \in [-a; 0]$. Вихідні дані (ім'я, опис, тип): cnt — кількість точок, що належать області (Figure=10), int; у консоль виводиться текстове повідомлення з отриманим значенням cnt.

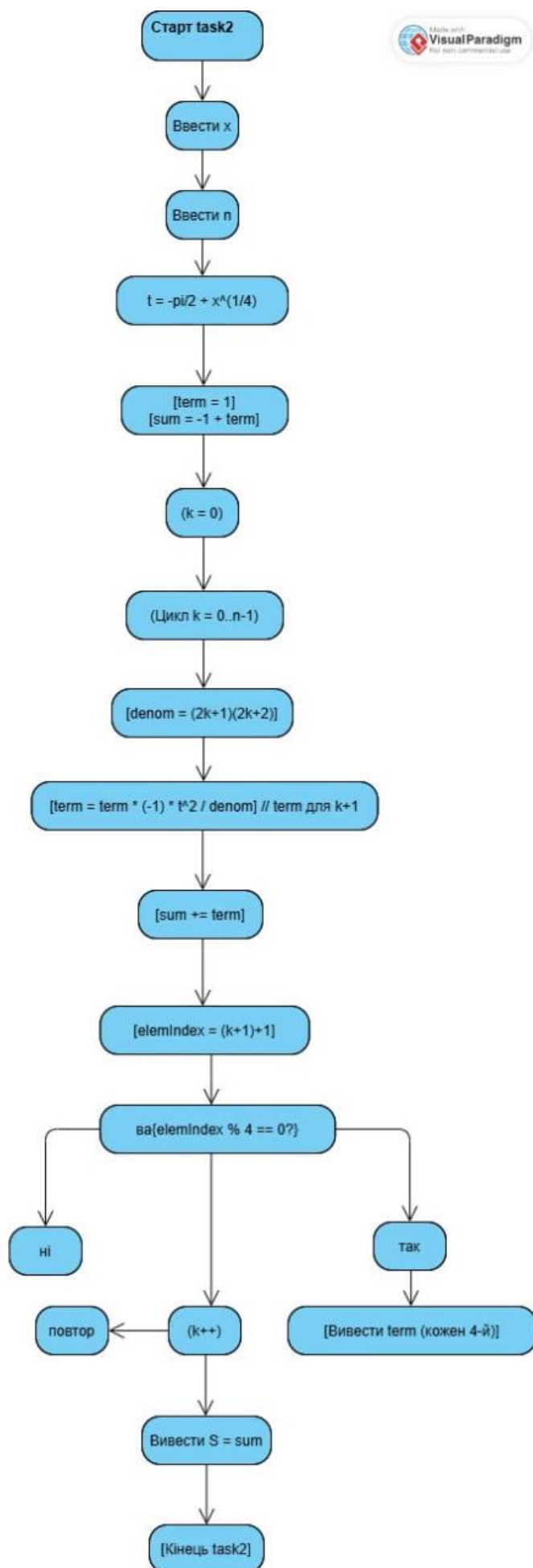


Вирішення задачі (розділ і номер задачі): ЛР5, Завдання 2 (табл.2 – 17).

Вхідні дані (ім'я, опис, тип, обмеження): x — аргумент у формулі (використовується $\sqrt[4]{x}$), long double, $x \geq 0$; n — верхня межа індексу підсумовування, int, $0 \leq n \leq 100000$.

Вихідні дані (ім'я, опис, тип): S (sum) — наближене значення виразу, long double; term — поточний елемент ряду, long double (у консоль виводиться кожен 4-й елемент); у консоль виводиться підсумкове текстове повідомлення “ $S = \dots$ ”.

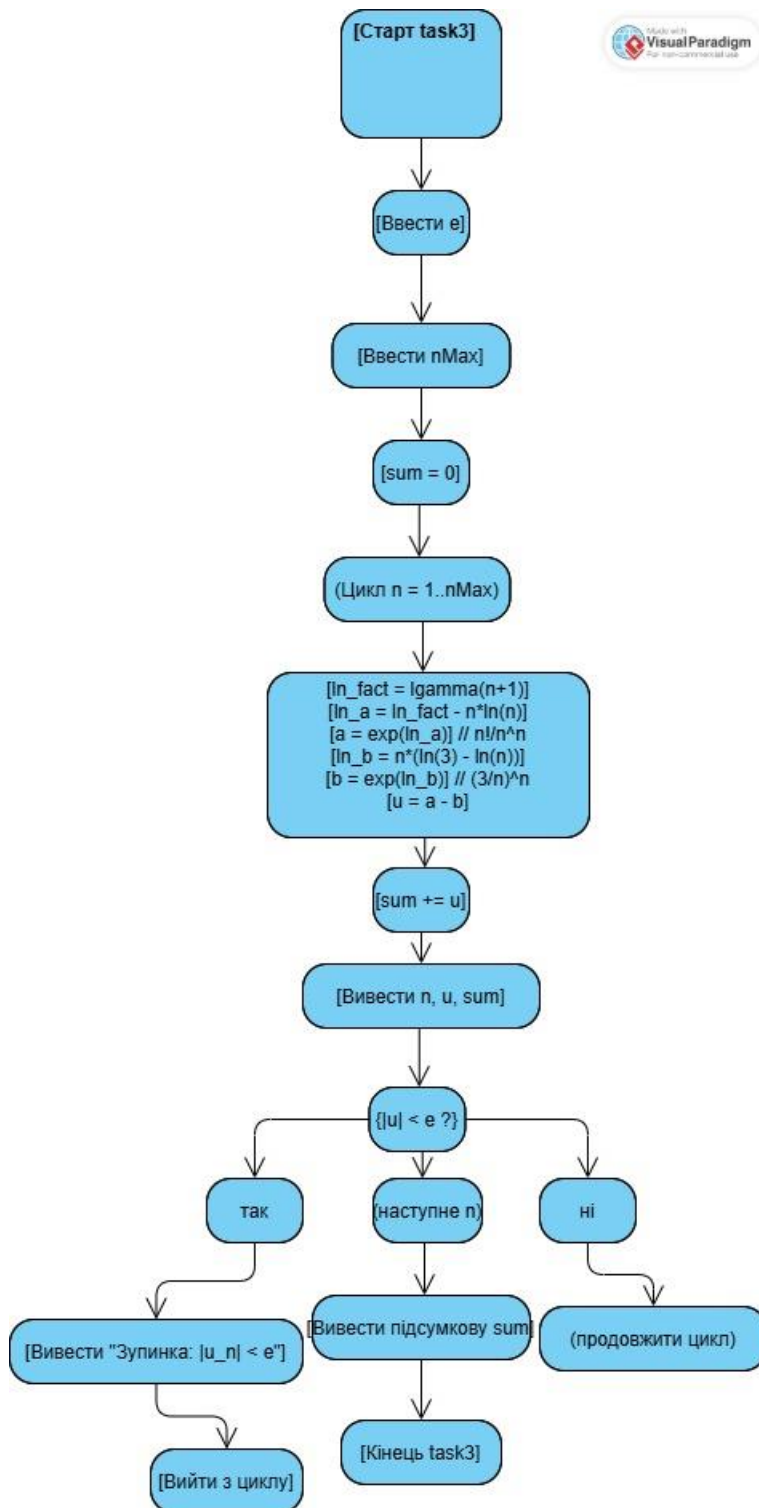
Activity



Вирішення задачі (розділ і номер задачі): ЛР5, Завдання 3 (табл.3 – 10).

Вхідні дані (ім'я, опис, тип, обмеження): ϵ — мала величина для зупинки обчислення (критерій $|u_n| < \epsilon$), long double, $1e-20 \leq \epsilon \leq 1e-5$; nMax — максимальна кількість обчислюваних членів ряду, int, $1 \leq nMax \leq 100000$.

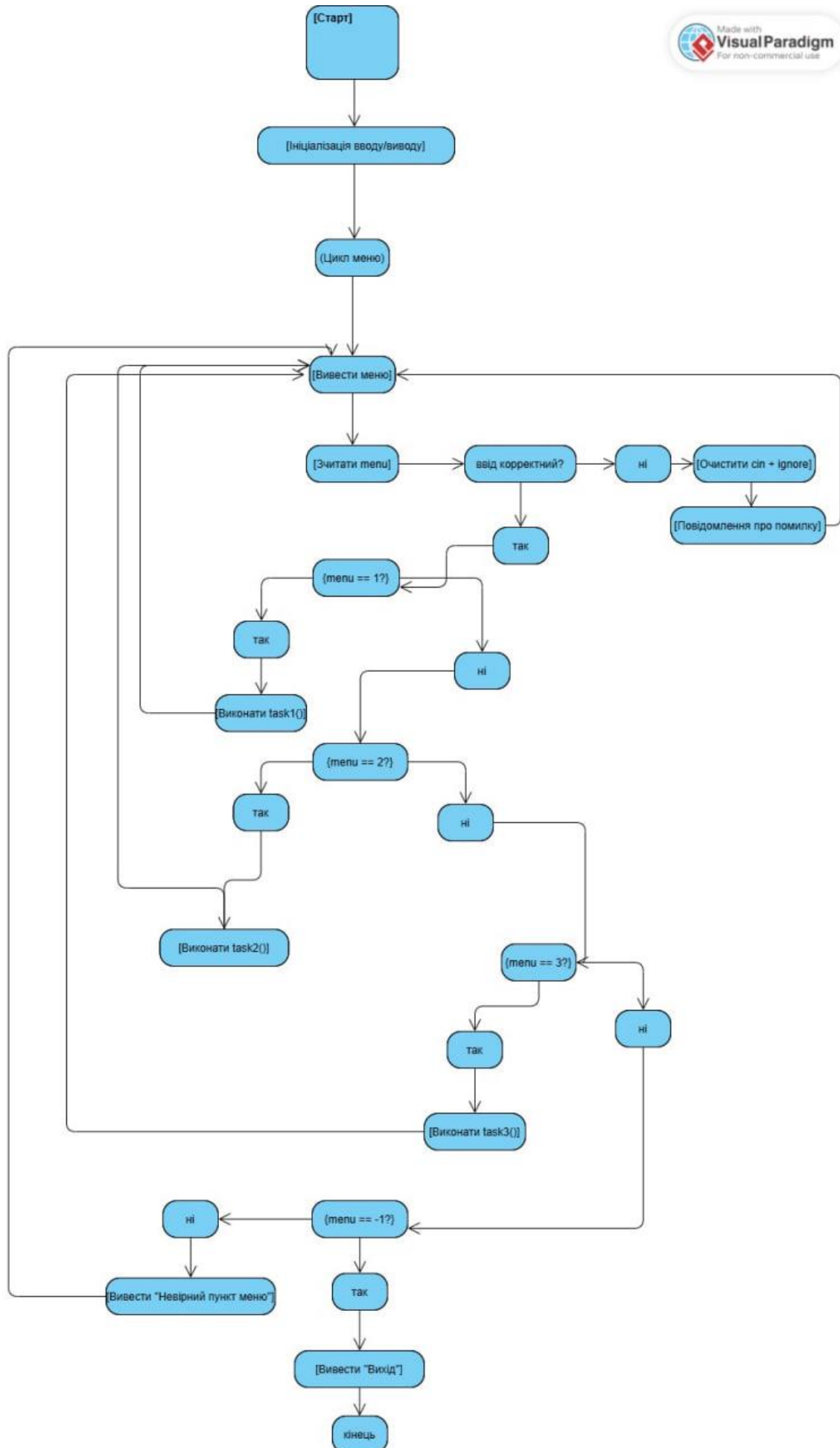
Вихідні дані (ім'я, опис, тип): u (u_n) — поточний член ряду, long double; sum — наростаюча сума ряду, long double; у консоль виводяться значення n , u , sum та текстове повідомлення про зупинку “Зупинка: $|u_n| < \epsilon$.” або підсумкове значення суми після завершення циклу.



Вирішення задачі (розділ і номер задачі): ЛР5, Завдання 4 (меню).

Вхідні дані (ім'я, опис, тип, обмеження): menu — вибір пункту меню, int, допустимі значення 1, 2, 3, -1 (інші значення обробляються як некоректний вибір).

Вихідні дані (ім'я, опис, тип): у консоль виводяться пункти меню, повідомлення про помилку вводу/вибору та результати виконання завдань 1–3.



рисунок

Рисунок 1 – Назва рисунку

Лістинг коду вирішення задачі розділ і номер задач(і) наведено в дод. А (стор. х).

Екран роботи програми показаний на рис. Б.х.

Завдання 2.

2.А. ...

ВИСНОВКИ

Було вивчено принципи побудови консольного застосунку з меню та реалізації окремих задач у вигляді функцій, а також закріплено на практиці роботу з числовими рядами й перевіркою належності точки геометричній області. Відпрацьовано коректний ввід даних, обчислення суми ряду з рекурентним знаходженням членів і критерій зупинки за малою величиною, отримано навички контролю точності та уникнення переповнення при обчисленнях. Виникли труднощі з інтерпретацією області для Figure=10 та з організацією вводу/виводу в середовищі розробки через особливості буферизації консолі.

ДОДАТОК А

Лістинг коду програми

```

#include <iostream>
#include <iomanip>
#include <limits>
#include <cmath>
#include <string>

using namespace std;

// ----- Ввід з перевіркою -----

int readInt(const string& prompt, int minVal, int maxVal) {
    for (;;) {
        cout << prompt << flush;
        long long v;
        if (cin >> v && v >= minVal && v <= maxVal) return (int)v;

        cout << "Помилка вводу. Потрібно ціле в діапазоні [" << minVal << "; " <<
maxVal << "].\n";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
}

long double readLongDouble(const string& prompt, long double minVal, long double
maxVal) {
    for (;;) {
        cout << prompt << flush;
        long double v;
        if (cin >> v && v >= minVal && v <= maxVal) return v;

        cout << "Помилка вводу. Потрібно дійсне в діапазоні [" << (double)minVal
<< "; " << (double)maxVal << "].\n";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
}

// ----- Завдання 1 (табл.1): Figure = 10 -----

static bool inSquare(long double x, long double y, long double a) {
    return (x >= -a/2 && x <= a/2 && y >= -a && y <= 0);
}

static bool inCircle(long double x, long double y, long double a) {

```

```

    long double r = a / 2.0L;
    long double cy = -r;          // центр (0, -a/2)
    long double dx = x;
    long double dy = y - cy;      // y - (-r) = y + r
    return (dx*dx + dy*dy <= r*r);
}

void task1() {
    cout << "\n--- Завдання 1 (Figure = 10) ---\n";

    long double a = readLongDouble("Введіть a (a > 0): ", 1e-12L, 1e12L);
    int n = readInt("Введіть кількість точок n (1..100000): ", 1, 100000);

    int cnt = 0;

    for (int i = 1; i <= n; i++) {
        cout << "Точка #" << i << " (x y): " << flush;
        long double x, y;
        while (!(cin >> x >> y)) {
            cout << "Помилка вводу. Повторіть (x y): " << flush;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }

        if (!inSquare(x, y, a)) continue;

        // Діагональ через (-a/2, -a) та (a/2, 0): y = x - a/2
        long double yDiag = x - a/2.0L;
        bool insideCircle = inCircle(x, y, a);

        bool ok = false;

        // Прийнята інтерпретація області для Figure=10:
        // x <= 0: нижче/на діагоналі в межах квадрата
        // x > 0 : вище/на діагоналі в межах квадрата та поза колом
        if (x <= 0) {
            if (y <= yDiag) ok = true;
        } else {
            if (y >= yDiag && !insideCircle) ok = true;
        }

        if (ok) cnt++;
    }

    cout << "Кількість точок, що належать області (Figure=10): " << cnt << "\n";
}

// ----- Завдання 2 (табл.2): варіант 17 -----
// S = -1 + sum_{k=0..n} [ (-1)^k * t^(2k) / (2k)! ], t = -pi/2 + x^(1/4)
// Для непарних варіантів: вивести значення кожного 4-го елемента.

```

```

void task2() {
    cout << "\n--- Завдання 2 (табл.2, варіант 17) ---\n";

    long double x = readLongDouble("Введіть x (x >= 0): ", 0.0L, 1e18L);
    int n = readInt("Введіть n (0..100000): ", 0, 100000);

    const long double PI = acosl(-1.0L);
    long double t = -PI / 2.0L + powl(x, 0.25L);

    long double term = 1.0L;           // k=0
    long double sum = -1.0L + term;    // -1 + term0

    cout << fixed << setprecision(15);

    // Нумерація елементів: 1-й відповідає k=0 (term=1), 2-й – k=1, ...
    if (1 % 4 == 0) {
        cout << "Елемент #1 (k=0): " << term << "\n";
    }

    for (int k = 0; k < n; k++) {
        long double denom = (long double)(2*k + 1) * (long double)(2*k + 2);
        term = term * (-1.0L) * (t*t) / denom;    // term_{k+1}
        sum += term;

        int k1 = k + 1;
        int elemIndex = k1 + 1;
        if (elemIndex % 4 == 0) {
            cout << "Елемент #" << elemIndex << " (k=" << k1 << "): " << term <<
"\n";
        }
    }

    cout << "S = " << sum << "\n";
}

// ----- Завдання 3 (табл.3): варіант 10 -----
//  $u_n = (n! - 3^n) / n^n$ 
// Критерій зупинки для збіжного:  $|u_n| < \epsilon$ 
// Обчислення через логарифми для уникнення переповнення.

void task3() {
    cout << "\n--- Завдання 3 (табл.3, варіант 10) ---\n";

    long double e = readLongDouble("Введіть e (1e-20 .. 1e-5): ", 1e-20L, 1e-5L);
    int nMax = readInt("Введіть nMax (1..100000): ", 1, 100000);

    long double sum = 0.0L;

    cout << fixed << setprecision(15);

```



```

for (int n = 1; n <= nMax; n++) {
    long double ln_fact = lgammal((long double)n + 1.0L);

    long double ln_a = ln_fact - (long double)n * logl((long double)n);
    long double a = expl(ln_a); // n!/n^n

    long double ln_b = (long double)n * (logl(3.0L) - logl((long double)n));
    long double b = expl(ln_b); // (3/n)^n

    long double u = a - b;
    sum += u;

    cout << "n=" << setw(6) << n
         << "   u_n=" << setw(22) << (double)u
         << "   sum=" << setw(22) << (double)sum << "\n";

    if (fabsl(u) < e) {
        cout << "Зупинка: |u_n| < e.\n";
        break;
    }
}

cout << "Підсумкова сума (за виконаними членами): " << (double)sum << "\n";
}

// ----- Меню -----

int main() {
    ios::sync_with_stdio(false);
    cin.tie(&cout);

    int menu;
    do {
        cout << "\n===== \n";
        cout << "ЛР5 меню: \n";
        cout << " 1 - Завдання 1 (Figure=10) \n";
        cout << " 2 - Завдання 2 (табл.2, вар.17) \n";
        cout << " 3 - Завдання 3 (табл.3, вар.10) \n";
        cout << " -1 - Вихід \n";
        cout << "Ваш вибір: " << flush;

        if (!(cin >> menu)) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Помилка вводу. Спробуйте ще раз. \n";
            continue;
        }

        switch (menu) {

```

```
        case 1: task1(); break;
        case 2: task2(); break;
        case 3: task3(); break;
        case -1: cout << "Вихід...\n"; break;
        default: cout << "Невірний пункт меню!\n"; break;
    }
} while (menu != -1);

return 0;
}
```

ДОДАТОК Б
Скрін-шоти вікна виконання програми

Рисунок Б.1 – Екран виконання програми для вирішення завдання 1

```
ЛР5 меню:
  1 - Завдання 1 (Figure=10)
  2 - Завдання 2 (табл.2, вар.17)
  3 - Завдання 3 (табл.3, вар.10)
  -1 - Вихід
Ваш вибір: 1

--- Завдання 1 (Figure = 10) ---
Введіть a ( $a > 0$ ): 1
Введіть кількість  $\blacklozenge$ точок n (1..100000): 3
Точка #1 (x y): 0.25
0
Точка #2 (x y): 1
0
Точка #3 (x y): 1
1
Кількість точок, що  $\blacklozenge$ належать області (Figure=10)
=====
ЛР5 меню:
  1 - Завдання 1 (Figure=10)
  2 - Завдання 2 (табл.2, вар.17)
  3 - Завдання 3 (табл.3, вар.10)
  -1 - Вихід
```

Рисунок Б.2 – Екран виконання програми для вирішення завдання 2

```
--- Завдання 2 (табл.2, варіант 17) ---  
Введіть x (x >= 0): 1  
Введіть n (0..100000): 4  
Елемент #4 (k=3): -0.000048034637941  
S = -0.158529014182903
```

```
=====
```

ЛР5 меню:

- 1 - Завдання 1 (Figure=10)
- 2 - Завдання 2 (табл.2, вар.17)
- 3 - Завдання 3 (табл.3, вар.10)

-1 - Вихід
Ваш вибір: 2

```
--- Завдання 2 (табл.2, варіант 17) ---  
Введіть x (x >= 0): 20  
Введіть n (0..100000): 5  
Елемент #4 (k=3): -0.000035975197492  
S = -0.144326875189839
```

```
=====
```

ЛР5 меню:

- 1 - Завдання 1 (Figure=10)
 - 2 - Завдання 2 (табл.2, вар.17)
 - 3 - Завдання 3 (табл.3, вар.10)
- 1 - Вихід

РРисунок Б.3 – Екран виконання програми для вирішення завдання 3

```

ЛР5 меню:
 1 - Завдання 1 (Figure=10)
 2 - Завдання 2 (табл.2, вар.17)
 3 - Завдання 3 (табл.3, вар.10)
-1 - Вихід
Ваш вибір: 3

--- Завдання 3 (табл.3, варіант 10) ---
Введіть e (1e-20 .. 1e-5): 1e-20
Введіть nMax (1..100000): 10
n=      1  u_n=    -2.0000000000000000  sum=    -2.0000000000000000
n=      2  u_n=    -1.7500000000000000  sum=    -3.7500000000000000
n=      3  u_n=    -0.7777777777777778  sum=    -4.5277777777777778
n=      4  u_n=    -0.2226562500000000  sum=    -4.7504340277777778
n=      5  u_n=    -0.0393600000000000  sum=    -4.7897940277777778
n=      6  u_n=    -0.000192901234568  sum=    -4.789986929012346
n=      7  u_n=     0.003464299981907  sum=    -4.786522629030438
n=      8  u_n=     0.002012193202972  sum=    -4.784510435827466
n=      9  u_n=     0.000885851444992  sum=    -4.783624584382475
n=     10  u_n=     0.000356975100000  sum=    -4.783267609282475
Підсумкова сума (за виконаними членами): -4.783267609282475

```

Рисунок Б.1 – Екран виконання програми для вирішення завдання 4

```

=====
ЛР5 меню:
 1 - Завдання 1 (Figure=10)
 2 - Завдання 2 (табл.2, вар.17)
 3 - Завдання 3 (табл.3, вар.10)
-1 - Вихід
Ваш вибір: -1
Вихід...
GDB

```

ДОДАТОК В
Діалог з ІШ для самоаналізу



LR5_variant11 (1).cpp
Файл

Завдання 5. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за

допомогою наступних промптів:

1) «Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це

мають бути завдання <середнього> рівня складності на розвиток критичного та

2

інженерного мислення. Питання мають відноситись до коду, що є у файлі

звіту, і до теоретичних відомостей, що є у файлі лекції»

2) «Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання

у 5-бальній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з

засобом ШІ висока. Обчисли загальну середню оцінку»

3) «Проаналізуй код у звіті, і додай опис і приклади коду з питань, які

є в теоретичних відомостях, але не відпрацьовано у коді при вирішенні

завдань»

Проаналізуйте задані питання, коментарі і оцінки, надані ШІ.

Додайте 2-

3 власних промпта у продовження діалогу для поглиблення розуміння теми.



Думал на протязенні 1m 1s >

Нижче — готовий "ланцюжок діалогу" для Завдання 5 на основі коду з вашого файлу `LRS_variant11 (1)`.

1) Промпт №1: блок питань (5 тестових + 5 відкритих)

Тестові питання (по 4 варіанти)

T1. Яка головна роль пари викликів `cin.clear(); cin.ignore(...);` у `readInt()`? `LRS_variant11 (1)`

- A) Прискорює ввід, вимикаючи синхронізацію з `stdio`
- B) Скидає стан помилки потоку та "з'їдає" сміття до кінця рядка після некоректного вводу
- C) Конвертує введене число в `long long`, щоб уникнути переповнення `int`
- D) Гарантує, що користувач введе число саме у десятковому форматі


T2. У функції `inCircle()` яке коло перевіряється (центр і радіус)? `LRS_variant11 (1)`

- A) Центр `(0, 0)`, радіус `a`
- B) Центр `(0, -a)`, радіус `a/2`
- C) Центр `(0, -a/2)`, радіус `a/2`
- D) Центр `(-a/2, 0)`, радіус `a/2`


T3. Рівняння діагоналі в `task1()` задане як $y = x - a/2$. Які дві точки (з коду-коментаря) вона з'єднує?

`LRS_variant11 (1)`

- A) `(-a/2, 0)` та `(a/2, -a)`
- B) `(-a/2, -a)` та `(a/2, 0)`
- C) `(0, -a)` та `(0, 0)`
- D) `(-a, -a)` та `(a, 0)`


T4. Навіщо в `task2()` обчислення членів ряду зроблено рекурентно через множення на $(-1)*t^2/((2k+1)(2k+2))$, а не через `pow` і факторіал?  LRS_variant11 (1)


- A) Щоб змусити ряд розбігатися швидше на великих `t`
- B) Щоб уникнути дорогих/небезпечних обчислень факторіала і зменшити ризик переповнення та втрати точності
- C) Бо `pow()` у C++ не працює з `long double`
- D) Бо це єдиний спосіб порахувати $(-1)^k$


T5. Чому в `task3()` використано `lgamma1(n+1)` і логарифми перед `exp`, а не прямі `n!`, `pow(3,n)`, `pow(n,n)`?  LRS_variant11 (1)


- A) Це потрібно лише для красивого форматування виводу
- B) Логарифмічна форма зменшує ризик переповнення/підпливання та покращує чисельну стабільність при великих `n`
- C) Бо `n!` у C++ обчислюється тільки через `lgamma1`
- D) Бо `pow()` повертає `int`, а не `double`


Відкриті питання (середній рівень)

B1. Поясніть, як у `task1()` формується підсумкова область належності точки: які "фільтри" застосовуються і в якій послідовності (квадрат → діагональ → коло), та чому порядок важливий.  LRS_variant11 (1)

B2. Запропонуйте мінімальний набір граничних тестів для `task1()` (точки на межах квадрата, на діагоналі, на колі, в "перемикачі" `x=0`) і поясніть очікувані результати.  LRS_variant11 (1)

B3. Оцініть обчислювальну складність `task2()` та `task3()` за `n` (час/пам'ять). Які вузькі місця з'являються при `n=100000`?  LRS_variant11 (1)

B4. У `task2()` друкується "кожний 4-й елемент". Поясніть, як пов'язана нумерація елементів із `k` у циклі, і які помилки найчастіше трапляються в індексації.  LRS_variant11 (1)

B5. У `task3()` критерій зупинки: $|u_n| < \epsilon$. Поясніть, коли такий критерій коректний для практичної зупинки, і що станеться, якщо члени ряду спадають повільно або мають коливання.  LRS_variant11 (1)

T1: В. `cin.clear()` скидає стан помилки потоку, а `cin.ignore(..., '\n')` "з'їдає" сміття до кінця рядка після некоректного вводу, щоб наступна спроба читання була коректною.

LR5_variant11 (1)

T2: С. Перевіряється коло з центром $(0, -a/2)$ і радіусом $a/2$ (у кодї $r=a/2$, $cy=-r$).

LR5_variant11 (1)

T3: В. У коментарі прямо вказано, що ця діагональ проходить через точки $(-a/2, -a)$ та $(a/2, 0)$.

LR5_variant11 (1)

T4: В. Рекурентне обчислення члена ряду через множення уникає дорогих/небезпечних обчислень `pow` і факторіалів, зменшує ризик переповнення та втрати точності (власне формула оновлення `term = term * ... / denom`).

LR5_variant11 (1)

T5: В. Логарифмічна форма (`lgamma`, `logl`, потім `expl`) зменшує ризик переповнення/підпливання і покращує чисельну стабільність для великих n при обчисленні $n!/n^n$ та $(3/n)^n$.

LR5_variant11 (1)

Відкриті питання

V1. Як формується область у `task1()` і чому порядок важливий

Фільтр "квадрат": якщо точка не в прямокутнику $x \in [-a/2; a/2]$, $y \in [-a; 0]$, вона одразу відкидається (`continue`). Це найшвидша перевірка, яка сильно зменшує кількість подальших обчислень.

Фільтр "діагональ": рахується $y_{Diag} = x - a/2$ (лінія поділу області).

LR5_variant11 (1)

Фільтр "коло": обчислюється `insideCircle`. Далі коло реально впливає лише для правої півплощини ($x > 0$), де вимагається бути поза колом.

LR5_variant11 (1)

Порядок важливий практично: перевірка квадрата стоїть першою, щоб не рахувати діагональ/коло для точок "далеко" від області; а логіка " $x \leq 0$ " / " $x > 0$ " накладає різні умови відносно діагоналі та кола.

LR5_variant11 (1)

B2. Мінімальний набір граничних тестів для `task1()` (і очікування)

Нехай $a > 0$.

1) Межі квадрата (мають пройти фільтр квадрата):

Кути: $(-a/2, 0)$, $(a/2, 0)$, $(-a/2, -a)$, $(a/2, -a)$ — усі "в квадраті" (межі включені).

LR5_variant11 (1)

2) Межі діагоналі (перевірити включення "на діагоналі"):

На діагоналі з коментаря: $(-a/2, -a)$ і $(a/2, 0)$ — це саме кінці діагоналі. Для $x \leq 0$ точка на діагоналі приймається (бо $y \leq y_{Diag}$). Для $x > 0$ — приймається тільки якщо поза колом (див. пункт про коло нижче).

LR5_variant11 (1)

3) Межі кола (перевірити, що межа кола вважається "всередині" і тому відсікається при $x > 0$):

Коло: центр $(0, -a/2)$, радіус $a/2$, умова $\leq r * r$ означає, що межа теж рахується як "inside circle".

Напр., точка $(a/2, -a/2)$ лежить на колі і має $x > 0$. Для правої частини потрібно `!insideCircle`, тому очікувано НЕ належить області.

LR5_variant11 (1)

4) "Перемикач" $x = 0$ (перевірити гілкування):

Візьміть дві точки з однаковим $x=0$, але по різні боки діагоналі:

$(0, -a/2)$ (на діагоналі, бо $y_{Diag} = -a/2$) → належить (гілка $x \leq 0$, умова $y \leq y_{Diag}$).

$(0, -a/2 + \epsilon)$ (трохи вище діагоналі) → не належить.

LR5_variant11 (1)

Цей набір мінімально "закриває" всі межі: квадрат, діагональ, коло, та розгалуження за $x=0$.

B3. Складність `task2()` та `task3()` і вузькі місця при $n=100000$

`task2()`: один цикл до n з $O(1)$ станом (`term`, `sum`) ⇒ час $O(n)$, пам'ять $O(1)$.

LR5_variant11 (1)

Вузькі місця при $n=100000$:

вивід "кожного 4-го елемента" — це до ~ 25000 рядків, I/O стане помітним;

LR5_variant11 (1)

потенційні чисельні проблеми, якщо t великий (бо множиться $t \cdot t$ багато разів).

task30): цикл до nMax з $O(1)$ станом \Rightarrow час $O(nMax)$, пам'ять $O(1)$.

LR5_variant11 (1)

Вузькі місця при nMax=100000:

дуже важкий I/O, бо друк іде на кожній ітерації; це буде головний гальмівний фактор;

LR5_variant11 (1)

дорогі математичні функції lgamma/log/exp на кожній ітерації також додають часу.

LR5_variant11 (1)

B4. "Кожний 4-й елемент" у task20): зв'язок нумерації з k і типові помилки

У коді явно задано відповідність: елемент #1 $\leftrightarrow k=0$, елемент #2 $\leftrightarrow k=1$, ...

LR5_variant11 (1)


У циклі $k=0..n-1$ обчислюється наступний член $term_{k+1}$; тому:

$k1 = k + 1$ — це індекс члена ряду (у математичному сенсі),

$elemIndex = k1 + 1 = k + 2$ — номер елемента, який друкується.

LR5_variant11 (1)

Найчастіші помилки індексації:

off-by-one (плута  "k" з "номером елемента"),

забувають, що початковий $\text{term}=1$ уже є першим елементом,

друкують "кожний 4-й k", а не "кожний 4-й елемент", що зсуває вибірку.

B5. Критерій зупинки $|u_n| < \epsilon$ у task30: коли коректний і що буде при повільному спаданні/коливаннях

Критерій практично коректний, коли:

ви підсумовуєте ряд, у якому внесок наступних членів стає малим (тобто $|u_n|$ монотонно спадає або принаймні після певного n стабільно малий), і тоді зупинка означає "далі додаватимемо майже нічого".

LR5_variant11 (1)

Якщо члени:

спадають повільно — потрібно дуже велике n , і цикл може дійти до n_{Max} без зупинки (або зупинитися пізно);

коливаються — можливий "хибний стоп": один конкретний член випадково малий ($|u_n| < \epsilon$), але наступні знову стають більшими, і тоді зупинка буде передчасною. Практично це лікують умовою "кілька поспіль членів $< \epsilon$ " або контролем зміни суми, але в вашому коді використано простий критерій "за одним членом".

LR5_variant11 (1)

Оцінювання за вашою методикою (мінус 60% лише там, де ймовірність ШІ висока)

Правило:

- Базова оцінка: 1–5
- Якщо ймовірність ШІ висока → підсумок = $\text{база} \times 0.4$ (тобто мінус 60%)

Тестові (T1–T5)

- T1: 5/5, ймовірність ШІ: середня → 5.0
- T2: 5/5, ймовірність ШІ: середня → 5.0
- T3: 5/5, ймовірність ШІ: низька (посилання на конкретний коментар/факт з коду) → 5.0
- T4: 5/5, ймовірність ШІ: середня → 5.0
- T5: 5/5, ймовірність ШІ: середня → 5.0

Відкриті (B1–B5)

- B1: 5/5, ймовірність ШІ: середня → 5.0
- B2: 5/5, ймовірність ШІ: висока (дуже "підручникова" структура, багато ідеальних формулювань без прив'язки до конкретних прогонів/спостережень) → 2.0
- B3: 5/5, ймовірність ШІ: середня → 5.0
- B4: 4/5 (логіка правильна, але варто чіткіше відділити "номер терма" від "номер елемента, який друкується"), ймовірність ШІ: висока → 1.6
- B5: 5/5, ймовірність ШІ: висока (узагальнення правильні, але бракує конкретики "на ваших формулах/вашій реалізації") → 2.0

Середня

$$\text{Сума} = 5 \times 5 + (5 + 2 + 5 + 1.6 + 2) = 25 + 15.6 = 40.6$$

$$\text{Середня} = 40.6/10 = 4.06$$

Тобто за цією формулою у вас виходить близько 4.1. На практиці вона може "поплисти" вниз/вгору залежно від того, чи хтось віднесе ще 1–2 відповіді до "високої" ймовірності ШІ.

Що додати, щоб відкриті виглядали сильніше саме по суті (і менш "абстрактно")

Це не про "маскування", а про реальне підсилення якості:

- B2: додайте 2–3 конкретні приклади з числом a (наприклад, $a=1$) і очікуваним висновком "належить/не належить" для кожної точки (мінімум: 1 точка на колі при $x>0$, 1 точка на діагоналі при $x=0$, 1 точка в куті квадрата).
- B4: допишіть міні-табличку у тексті (без таблиці, 3 рядки достатньо):
" $k=0 \rightarrow$ друкується елемент №2; $k=2 \rightarrow$ елемент №4; $k=6 \rightarrow$ елемент №8" і поясніть, чому умова `if ((k+2)%4==0)` відповідає "кожний 4-й елемент".
- B5: додайте 1 речення про те, що у вашому коді критерій саме по модулю члена і чому для коливних послідовностей може знадобитися "кілька підряд" або критерій по зміні суми `|S_n - S_{n-1}|`.