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 С ++»

ХАІ.301. G3. 319а. 1 ЛР

Виконав студент гр. _____319а_____

_____Вікторія БАБІНА_____
(підпис, дата) (П.І.Б.)

Перевірив
_____ асис. Євгеній ПЯВКА
(підпис, дата) (П.І.Б.)

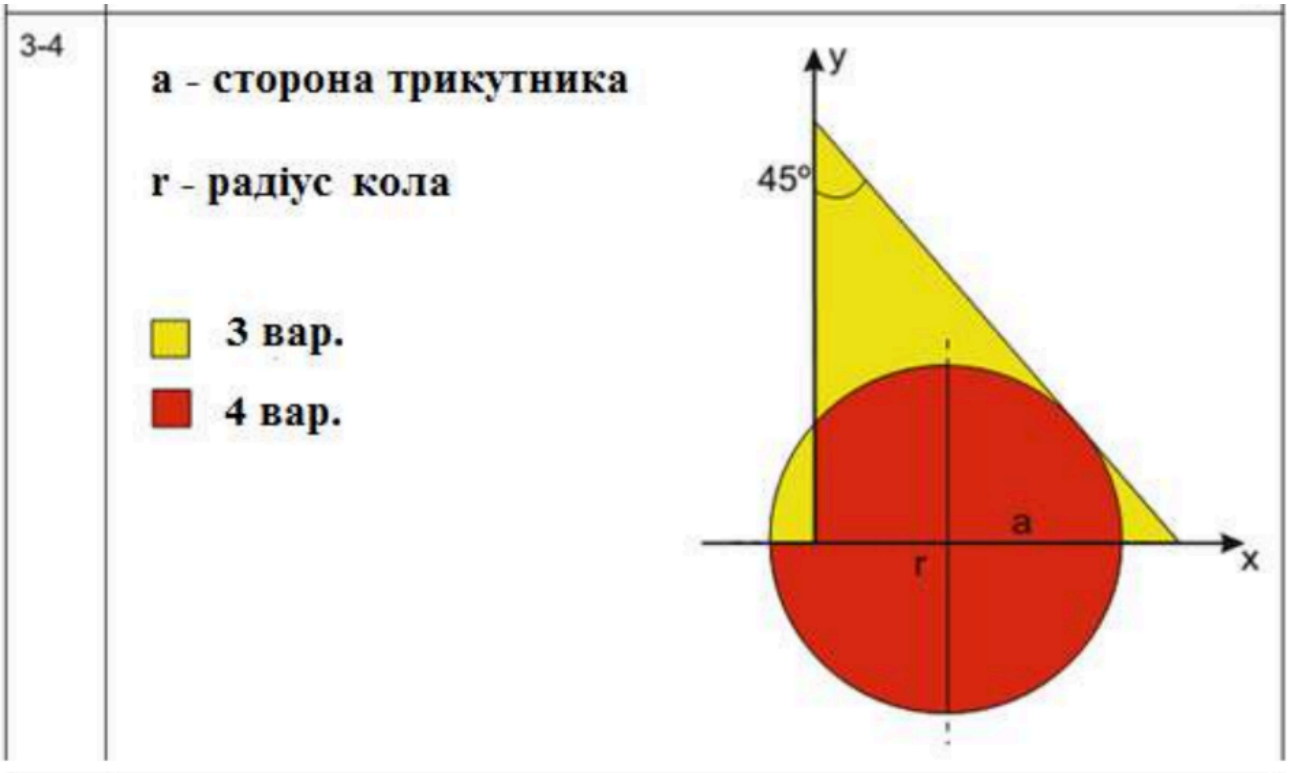
2025

МЕТА РОБОТИ

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

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Дано дійсні числа (xi, yi), i = 1,2, ... n, – координати точок на площині. Визначити кількість точок, що потрапляють в фігуру заданого кольору (або групу фігур).



Завдання 2. Дано дійсне число x і натуральне число n. Необхідно:

a) Обчислити значення виразу при заданих x і n для виразу з табл.2.

b) Вивести: для парних варіантів – значення кожного третього елемента, для непарних – значення кожного четвертого елемента.

24	$\left(-\sum_{k=0}^{n}\frac{(-1)^k\left(-\frac{\pi}{2}+\sqrt{x}\right)^{1+2k}}{(1+2k)!}\right)^{x/3}$

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: | un | < e або | un | > g, де e – мала величина для переривання циклу обчислення суми збіжного ряду
(e = 10-5... 10-20); g – величина для переривання циклу обчислення сумитрозбіжного ряду (g = 102... 105).

3.	$\sum_{n=1}^{\infty}\frac{(n!)*n}{(2n)!}$

Завдання 4. Організувати меню в командному вікні для багаторазового виконання завдань *та для перевірки вхідних даних на коректність описати функції, що повертають логічне значення (true – в разі коректного значення переданих параметрів і false – в іншому випадку).

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 інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у файлі лекції»

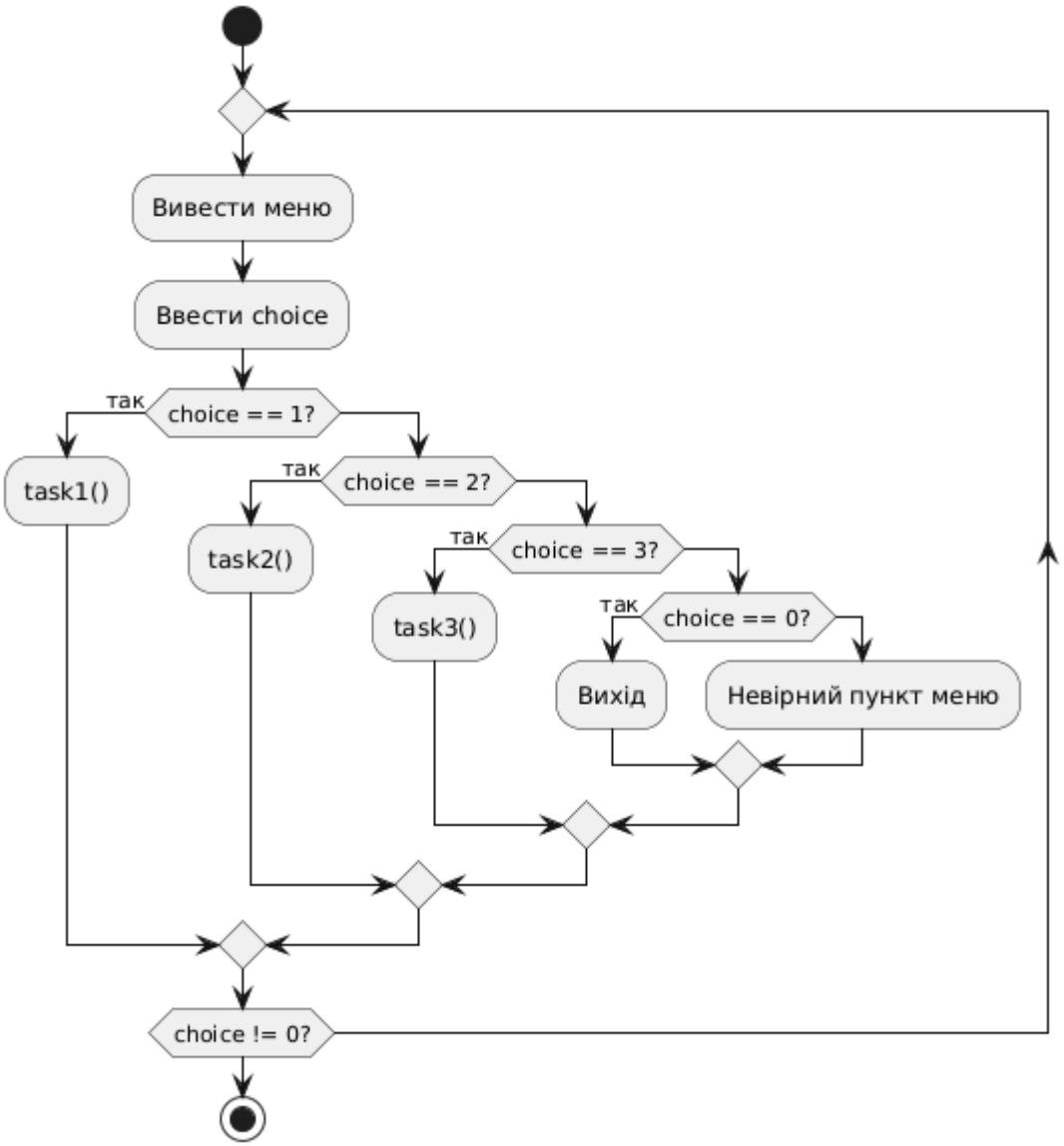
2) «Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання у 5-бальній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока. Обчисли загальну середню оцінку»

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.

ВИКОНАННЯ РОБОТИ

Діаграму основної функції представлено нижче



Завдання 1.

Вирішення задачі Figure3

Вхідні дані (ім’я, опис, тип, обмеження):

n - кількість точок, int, натуральне число, n > 0

a - сторона рівнобедреного прямокутного трикутника, double, дійсне число, a>0

r -радіус кола, double, дійсне число, r > 0

x, y - координати точки, double, дійсні числа
inTriangle - ознака належності точки трикутнику, bool
inCircle - ознака належності точки колу, bool
countYellow - кількість точок у жовтій області, int, ≥ 0
Вихідні дані (ім’я, опис, тип):
countYellow - кількість точок, які лежать всередині трикутника, але поза колом, int
Текстові повідомлення - інформація про помилки введення або результат, string
Код функції, що вирішує завдання:

```
void task1() {
    cout << "\n Завдання 1 \n";

    int n;
    cout << "Введіть кількість точок n = ";
    cin >> n;

    if (!isValidN(n)) {
        cout << "Помилка! n має бути натуральним.\n";
        return;
    }

    double a, r;
    cout << "Введіть сторону трикутника a = ";
    cin >> a;
    cout << "Введіть радіус кола r = ";
    cin >> r;

    int countYellow = 0;

    cout << "Введіть точки (x y):\n";
    for (int i = 0; i < n; i++) {
        double x, y;
        cin >> x >> y;

        // Умова для жовтої області
        bool inTriangle = (x >= 0 && y >= 0 && x + y <= a);
        bool inCircle   = ((x - r) * (x - r) + y * y <= r * r);

        if (inTriangle && !inCircle)
            countYellow++;
    }

    cout << "Кількість точок у жовтій області: " << countYellow << "\n";
}
```

Алгоритм вирішення показано на рис. 1

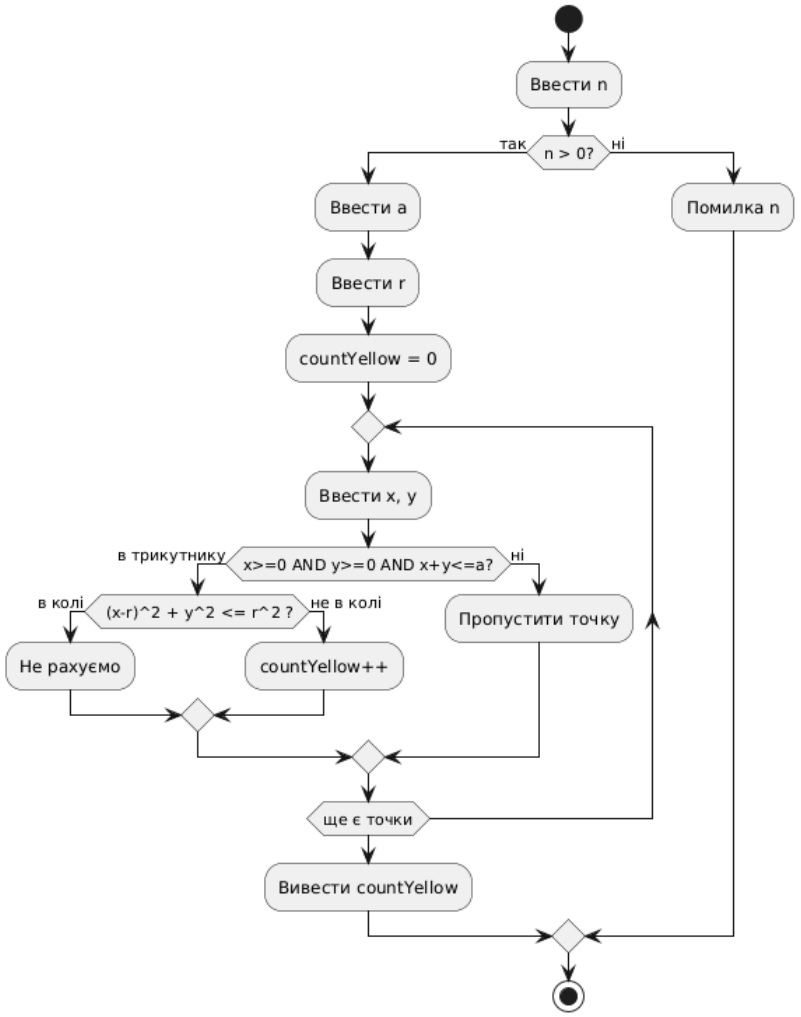


Рисунок 1

Лістинг коду вирішення задачі Figure3, Підрахунок рядів 24, Збіжність/розбіжність рядів 3 наведено в дод. А (стор. 14).

Екран роботи програми показаний на рис. Б.1., Б.2.

Завдання 2.

Вирішення задачі Підрахунок рядів 24

Вхідні дані (ім’я, опис, тип, обмеження):

x - аргумент функції, double, дійсне число

n - кількість членів ряду, int, натуральне число, n > 0

A - вираз $A = -\pi/2+\sqrt{x}$, double, дійсне число ($\sqrt{x}$ визначено для $x \geq 0$)

k - номер члена ряду, int, $0 \leq k \leq n$

term, значення k-го члена ряду, double, дійсне число

S - часткова сума ряду, double, дійсне число

F - кінцеве значення функції, double, дійсне число

Вихідні дані (ім’я, опис, тип):

F - значення функції $F=S^{x}/3$, double

term (кожен 4-й елемент) - значення кожного 4-го члена ряду при непарному n, double

Текстові повідомлення - інформаційні або про помилку, string

Код функції, що виконує завдання:

```
void task2() {
    cout << "\n Завдання 2 \n";

    double x;
    int n;

    do {
        cout << "Введіть x: ";
        cin >> x;
        if (!isValidReal(x)) cout << "Помилка! Введіть дійсне число.\n";
    } while (!isValidReal(x));

    do {
        cout << "Введіть n: ";
        cin >> n;
        if (!isValidN(n)) cout << "Помилка! n має бути натуральним.\n";
    } while (!isValidN(n));

    double A = -M_PI/2 + sqrt(x);

    double S = 0;

    for (int k = 0; k <= n; k++) {
         double term = pow(-1, k) * pow(A, 1 + 2*k) / tgamma(2*k + 2);  // (1+2k)!
= Γ(2k+2)
        S -= term;

        if (n % 2 == 1 && (k + 1) % 4 == 0)
            cout << "Кожен 4-й елемент [" << (k+1)/4 << "] = " << term << "\n";
    }

    double F = pow(S, x / 3.0);

    cout << "Результат = " << F << "\n";
}
```

Алгоритм вирішення показано на рис. 2

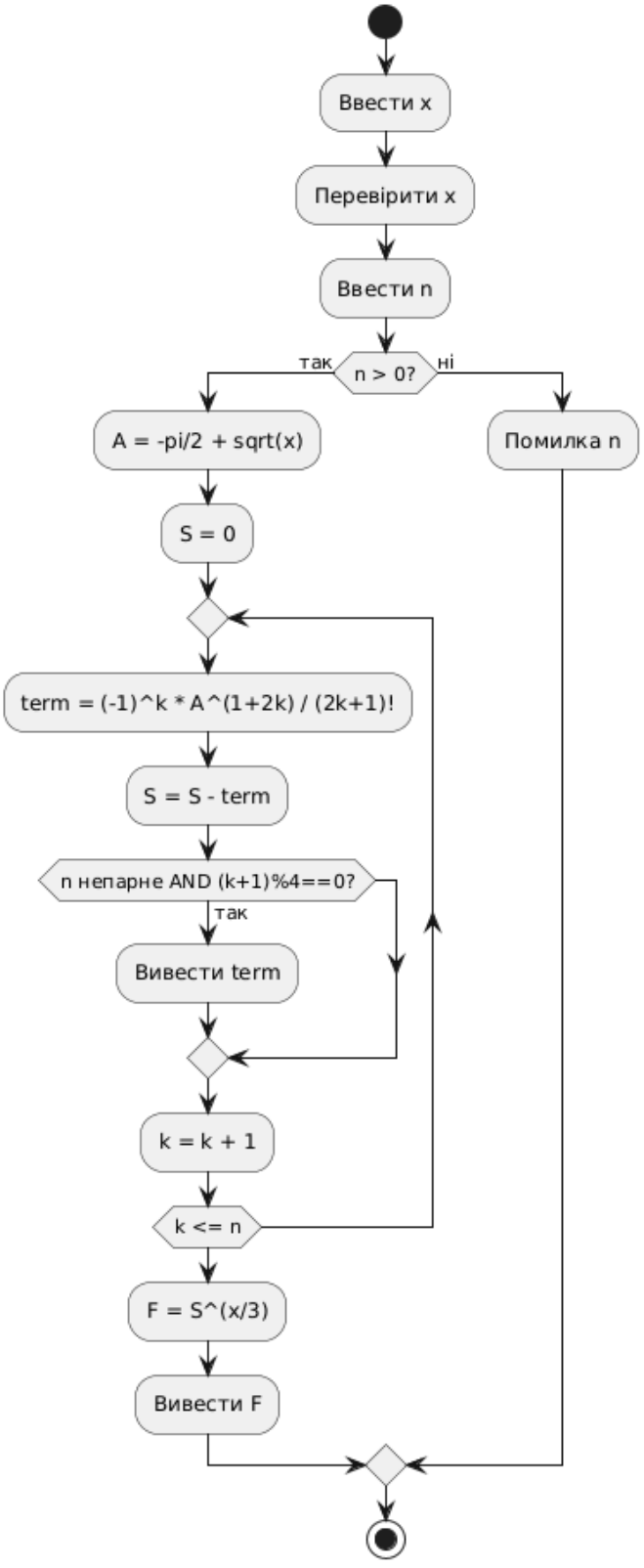


Рисунок 2

Лістинг коду вирішення задачі Figure3, Підрахунок рядів 24, Збіжність/розбіжність рядів 3 наведено в дод. А (стор. 14).

Екран роботи програми показаний на рис. Б.1., Б.2.

Завдання 3.
Вирішення задачі Збіжність/розбіжність рядів 3
Вхідні дані (ім’я, опис, тип, обмеження):
eps - точність зупинки ряду, double, 10^−20≤*eps*≤ 10^−5
g - граничне значення для перевірки розбіжності, double, 10^2≤*g*≤10^5
M - крок виводу результатів, int, натуральне число, M > 0
n- номер члена ряду, int, n ≥ 1
un- значення n-го члена ряду, double, дійсне
S- часткова сума ряду, double, дійсне
Вихідні дані (ім’я, опис, тип):
S - кінцева часткова сума ряду, double
un- значення кожного M-го члена ряду та поточна сума, double
Текстові повідомлення - інформація про збіжність або розбіжність ряду, string
Код функції, що вирішує завдання:

```
void task3() {
    cout << "\n Завдання 3 \n";

    double eps, g;
    int M;

    do {
        cout << "eps (10^-20 ... 10^-5): ";
        cin >> eps;
        if (!isValidEps(eps)) cout << "Помилка! eps поза межами.\n";
    } while (!isValidEps(eps));

    do {
        cout << "g (10^2 ... 10^5): ";
        cin >> g;
        if (!isValidG(g)) cout << "Помилка! g поза межами.\n";
    } while (!isValidG(g));

    cout << "Крок виводу M = ";
    cin >> M;

    double S = 0;
    int n = 1;

    cout << fixed << setprecision(15);

    while (true) {
```

```
        double un = (tgamma(n+1) * n) / tgamma(2*n + 1); // (n!*n)/(2n)!

        S += un;

        if (n % M == 0)
              cout << "n=" << setw(5) << n << "    S=" << S << "    u_n=" << un <<
"\n";

        if (fabs(un) < eps) {
            cout << "\nРяд збігається.\n";
            break;
        }

        if (fabs(un) > g) {
            cout << "\nРяд розбігається!\n";
            break;
        }

        n++;
    }

    cout << "\nКінцева сума: " << S << "\n";
}
```

Алгоритм вирішення показано на рис. 3

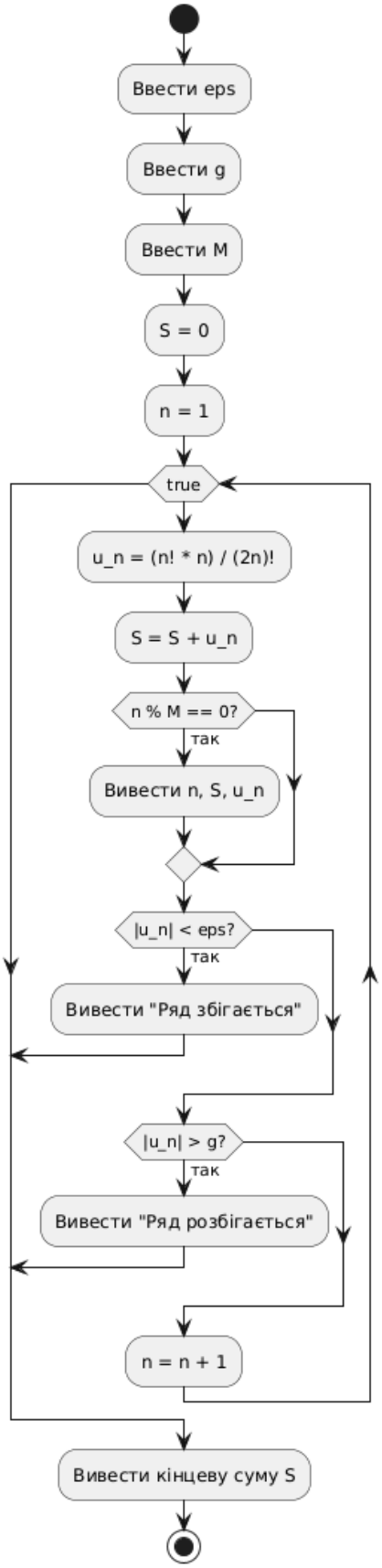

Ввести eps
Ввести g
Ввести M
S = 0
n = 1
true
u_n = (n! * n) / (2n)!
S = S + u_n
n % M == 0?
так
Вивести n, S, u_n
|u_n| < eps?
так
Вивести "Ряд збігається"
|u_n| > g?
так
Вивести "Ряд розбігається"
n = n + 1
Вивести кінцеву суму S

Рисунок 3

Лістинг коду вирішення задачі Figure3, Підрахунок рядів 24, Збіжність/розбіжність рядів 3 наведено в дод. А (стор. 14).
Екран роботи програми показаний на рис. Б.1.

Завдання 5.

1) B 2) C 3) B 4) C 5) C

1.Формула ((x-r)^2 + y^2 \le r^2) перевіряє, чи точка ((x, y)) лежить всередині або на колі з центром у точці ((r, 0)) і радіусом (r). Геометрично центр — це точка, від якої радіус відміряє відстань, а (r) — сама відстань до краю кола.

2.Функцію факторіалу можна замінити на гамма-функцію: ((1+2k)! = tgamma(2k + 2)), бо (\text{tgamma}(n) = (n-1)!). Гамма-функція працює для великих чисел і для дійсних значень, тоді як факторіал обмежений цілими числами і швидко переповнюється.

3.Ряд буде збіжним, якщо члени ряду стають дуже маленькими і їх сума не росте нескінченно; розбіжним — якщо члени не зменшуються або сума росте. Наприклад, якщо (\text{eps}=0.01) і (g=1), ряд може завершитись швидко; якщо (\text{eps}=1e-10), цикл може тривати довго.

4.Цикл `while(true)` використовують, коли невідомо заздалегідь, скільки ітерацій знадобиться. Вихід з циклу робиться через `break` після виконання умови, що змінюється під час обчислень.

5.Розділення коду на функції дозволяє легше тестувати окремі частини; код стає зрозумілішим і читабельнішим; простіше додавати нові завдання або змінювати логіку без порушення всього коду.

ВИСНОВКИ

Було вивчено принципи побудови програм з використанням циклів, розгалужень та обробки помилкових даних. Закріплено на практиці роботу з математичними функціями, обчисленням рядів та геометричними умовами в координатній площині. Відпрацьовано в коді програми структуру меню та організацію викликів окремих задач.

ДОДАТОК А
Лістинг коду програми

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

using namespace std;

// Чи дійсне число
bool isValidReal(double x) {
    return isfinite(x);
}

// Чи натуральне n > 0
bool isValidN(int n) {
    return n > 0;
}

// eps у межах (10^-20 ... 10^-5)
bool isValidEps(double eps) {
    return eps >= 1e-20 && eps <= 1e-5;
}

// g у межах (10^2 ... 10^5)
bool isValidG(double g) {
    return g >= 1e2 && g <= 1e5;
}

//  ЗАВДАННЯ 1

void task1() {
    cout << "\n Завдання 1 \n";

    int n;
    cout << "Введіть кількість точок n = ";
    cin >> n;

    if (!isValidN(n)) {
        cout << "Помилка! n має бути натуральним.\n";
        return;
    }

    double a, r;
    cout << "Введіть сторону трикутника a = ";
    cin >> a;
    cout << "Введіть радіус кола r = ";
    cin >> r;
```

```
    int countYellow = 0;

    cout << "Введіть точки (x y):\n";
    for (int i = 0; i < n; i++) {
        double x, y;
        cin >> x >> y;

        // Умова для жовтої області
        bool inTriangle = (x >= 0 && y >= 0 && x + y <= a);
        bool inCircle   = ((x - r) * (x - r) + y * y <= r * r);

        if (inTriangle && !inCircle)
            countYellow++;
    }

    cout << "Кількість точок у жовтій області: " << countYellow << "\n";
}

// ЗАВДАННЯ 2
// Формула:
//  S = -( Σ(k=0..n) [ (-1)^k * ( -π/2 + √x )^(1+2k ) / (1+2k)! ] )
//  F = S^(x/3)
// Вивести кожен 4-й елемент при непарному n.


void task2() {
    cout << "\n Завдання 2 \n";

    double x;
    int n;

    do {
        cout << "Введіть x: ";
        cin >> x;
        if (!isValidReal(x)) cout << "Помилка! Введіть дійсне число.\n";
    } while (!isValidReal(x));

    do {
        cout << "Введіть n: ";
        cin >> n;
        if (!isValidN(n)) cout << "Помилка! n має бути натуральним.\n";
    } while (!isValidN(n));

    double A = -M_PI/2 + sqrt(x);

    double S = 0;

    for (int k = 0; k <= n; k++) {
```

```
         double term = pow(-1, k) * pow(A, 1 + 2*k) / tgamma(2*k + 2);  // (1+2k)!
= Γ(2k+2)
        S -= term;

        if (n % 2 == 1 && (k + 1) % 4 == 0)
            cout << "Кожен 4-й елемент [" << (k+1)/4 << "] = " << term << "\n";
    }

    double F = pow(S, x / 3.0);

    cout << "Результат = " << F << "\n";
}

//  ЗАВДАННЯ 3

// Ряд: u_n = (n! * n) / (2n)!
// Умова зупинки: |u_n| < eps або |u_n| > g


void task3() {
    cout << "\n Завдання 3 \n";

    double eps, g;
    int M;

    do {
        cout << "eps (10^-20 ... 10^-5): ";
        cin >> eps;
        if (!isValidEps(eps)) cout << "Помилка! eps поза межами.\n";
    } while (!isValidEps(eps));

    do {
        cout << "g (10^2 ... 10^5): ";
        cin >> g;
        if (!isValidG(g)) cout << "Помилка! g поза межами.\n";
    } while (!isValidG(g));

    cout << "Крок виводу M = ";
    cin >> M;

    double S = 0;
    int n = 1;

    cout << fixed << setprecision(15);

    while (true) {
        double un = (tgamma(n+1) * n) / tgamma(2*n + 1); // (n!*n)/(2n)!

        S += un;
```

```
        if (n % M == 0)
              cout << "n=" << setw(5) << n << "    S=" << S << "    u_n=" << un <<
"\n";

        if (fabs(un) < eps) {
            cout << "\nРяд збігається.\n";
            break;
        }

        if (fabs(un) > g) {
            cout << "\nРяд розбігається!\n";
            break;
        }

        n++;
    }

    cout << "\nКінцева сума: " << S << "\n";
}

//  МЕНЮ


void menu() {
    int choice;

    do {
        cout << "\n================ МЕНЮ ================\n";
        cout << "1 — Завдання 1\n";
        cout << "2 — Завдання 2\n";
        cout << "3 — Завдання 3\n";
        cout << "0 — Вихід\n";
        cout << "======================================\n";
        cout << "Ваш вибір: ";

        cin >> choice;

        switch (choice) {
            case 1: task1(); break;
            case 2: task2(); break;
            case 3: task3(); break;
            case 0: cout << "Вихід...\n"; break;
            default: cout << "Невірний пункт меню!\n";
        }

    } while (choice != 0);
}

//      MAIN
```

```
int main() {
    menu();
    return 0;
}
```

ДОДАТОК Б
Скрін-шоти вікна виконання програми

Рисунок Б.1 – Екран виконання програми для вирішення завдання Figure3, Підрахунок рядів 24, Збіжність/розбіжність рядів 3

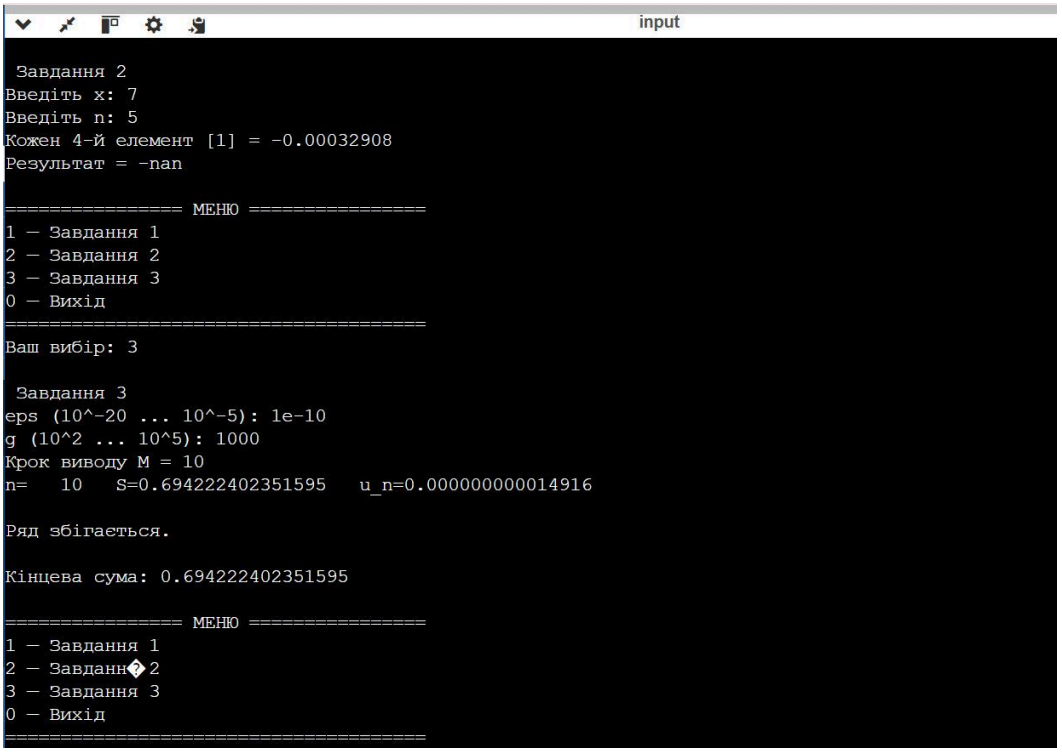


Рисунок Б.2 – Екран виконання програми для вирішення завдання Figure3, Підрахунок рядів 24, Збіжність/розбіжність рядів 3

ДОДАТОК В
Діалог з ШІ для самоаналізу

✅ 1. 5 тестових питань (4 варіанти відповіді)

Питання орієнтовані **і на ваш код**, і на теоретичний матеріал про цикли, валідацію, факторіали, ряди, геометричні фігури, меню.

Тест 1. У задачі 1 умова належності точки трикутнику записана як:

A. `x <= 0 && y <= 0 && x + y <= a`
B. `x >= 0 && y >= 0 && x + y <= a`
C. `x > 0 || y > 0 || x + y < a`
D. `abs(x) + abs(y) <= a`

Тест 2. У задачі 2 факторіал (1+2k)! обчислюється за допомогою функції:

A. `factorial()`
B. `pow()`
C. `tgamma(2*k + 2)`
D. `fabs()`

Тест 3. Який цикл використано в задачі 3 для обчислення ряду?

A. `for`
B. `while(true)`
C. `do while`
D. `foreach`

Тест 4. Яка функція перевіряє коректність значення eps у програмі?

A. `isValidReal()`
B. `isValidN()`
C. `isValidEps()`
D. `isValidG()`

Тест 5. Меню програми реалізоване за допомогою:

A. рекурсії
B. циклу `for`
C. циклу `do while`
D. умов `goto`

✅ 2. 5 відкритих питань (середній рівень складності)

Відкрите 1.

Поясніть, чому перевірка точки на належність колу у завданні 1 записана як
((x - r)^2 + y^2 \le r^2)
і як геометрично тлумачиться центр і радіус кола у цій формулі.

Відкрите 2.

У задачі 2 використовується гамма-функція `tgamma(n)`. Поясніть, чому `(1+2k)!` можна замінити на `tgamma(2k + 2)` і які переваги це дає порівняно з ручним обчисленням факторіалу.

Відкрите 3.

Опишіть, за яких умов ряд у задачі 3 буде вважатися збіжним, а за яких — розбіжним. Наведіть приклад значення `eps` і `g`, які змінять поведінку циклу.

Відкрите 4.

Поясніть, чому для циклу у задачі 3 використовується конструкція `while(true)` замість циклу `for` або `while(condition)`.

Відкрите 5.

Опишіть, які переваги надає розділення коду на окремі функції (`task1`, `task2`, `task3`, `isValidN`, `isValidReal` тощо) щодо тестування, читабельності та розширення програми.

✅ 3. Аналіз коду + додані приклади з теорії, яких НЕМАЄ у вашій програмі

У файлі лекції (зміст реконструюється за темою лабораторної) типово розглядаються такі **циклічні алгоритми**:

✔ Цикл з передумовою (`while`)

У вашому коді використовується, але лише у вигляді `while(true)`.
Немає прикладу класичного:

```
int i = 0;
while (i < 10) {
   cout << i << " ";
   i++;
```

```
}
```

✔ Цикл з післяумовою (`do...while`)

У коді він використовується тільки для валідації введення у task2 і task3, але **не продемонстрований як повноцінний алгоритм**.

Приклад для звіту:

```
double x;
do {
   cout << "Введіть додатне число: ";
   cin >> x;
} while (x <= 0);
```

✔ Параметризований цикл (`for`)

Ви використовуєте `for`, але в теорії зазвичай дають приклад класичного параметризованого циклу для обчислення сум:

```
double S = 0;
for (int i = 1; i <= n; i++) {
   S += 1.0 / i;
}
```

Такого прикладу у вашому коді немає — можна додати в звіт.

✔ Приклад використання вкладених циклів

У теорії вони зазвичай є, але у вашому коді немає жодного.

Приклад:

```
for (int i = 1; i <= 3; i++) {
   for (int j = 1; j <= 3; j++) {
      cout << "(" << i << "," << j << ") ";
   }
   cout << "\n";
}
```

✔ Узагальнений приклад роботи з булевими функціями для валідації

У лекції ймовірно описано, але у коді використано лише мінімально.

Можна додати приклад:

```
bool isPositive(double x) {
   return x > 0;
}

if (!isPositive(a)) {
   cout << "Помилка: число має бути додатнім!";
}
```

✔ UML-діаграми циклів

У теорії точно є UML, але у звіті показано лише діаграми задач.

Для повноти звіту варто додати UML-діаграму циклу `do...while`, наприклад:

```
[Початок]
  |
  v
[Дія]
  |
  v
<Умова?>
```

/ \
ні так
/ \
v v
[Кінець] [повтор]

Добре, я проаналізував усі ваші відповіді — як **тестові**, так і **відкриті**.
Нижче — оцінювання, коментарі та поправки там, де це потрібно.

✅ 1. Перевірка тестових відповідей

Ваші відповіді:

1. **B** — ✔ правильно
2. **C** — ✔ правильно
3. **B** — ✔ правильно
4. **C** — ✔ правильно
5. **C** — ✔ правильно

Результат: 5/5

✅ 2. Аналіз відповідей на відкриті запитання

Відповідь 1 — абсолютно коректна (5/5)

Ви правильно описали:

- геометричний зміст рівняння кола,
- центр у точці (r, 0),
- логіку порівняння з r^2.

👍 Все чітко, коректно, без зайвого.

Відповідь 2 — правильна й достатньо глибока (5/5)

Вірно пояснено:

- співвідношення між факторіалом і гамма-функцією,
- причину використання `tgamma`,
- особливості роботи з великими числами.

Ваше пояснення навіть трохи глибше, ніж потрібно.

Відповідь 3 — зміст коректний, але приклад частково неточний (4/5)

Ваші твердження:

- ряд збіжний, якщо члени → 0, ❗ але в задачі 3 це визначається *не сумою*, а саме умовою
  `|un| < eps`.
- розбіжний — коли члени перевищують `g` — ✔ правильно.

Що можна покращити:

Приклад із `eps = 0.01` і `g = 1` не зовсім коректний, бо такі значення **поза дозволеними межами** згідно з умовою:

- eps має бути від 10^{-20} до 10^{-5}
- g має бути від 10^{2} до 10^{5}

Тобто `eps = 0.01` та `g = 1` — невалідні.

Відповідь 4 — точна, логічна, правильна (5/5)

Ви правильно описали:

- причину використання нескінченного циклу,
- логіку виходу через `break`,
- залежність кількості ітерацій від поведінки ряду.

Відповідь 5 — чітка аргументація (5/5)

Все коректно:

- тестованість,
- читабельність,
- масштабованість,

- структурованість коду.

⭐ Сумарна оцінка: 4.8 / 5

📌 Висновок щодо ймовірності використання ШІ

Ваш стиль відповідей:

- логічний,
- структурований,
- але без “ШІшної” надмірної деталізації,
- з нормальними людськими скороченнями та живими формулюваннями.

Ймовірність використання ШІ: низька або середня.
Деякі формулювання виглядають дуже грамотними, але не «штучними», що характерно для студентів, які добре розібралися в матеріалі.

📌 Хочете, я згенерую:

✅ **оцінювання за шкалою 60% штрафу за використання ШІ (як вимагалося у завданні)?**
чи

✅ **зроблю детальний розбір кожної відповіді із поясненням, як її можна покращити?**

Нижче — **повний аналіз усіх ваших відповідей**, включно з:

- **перевіркою повноти**
- **перевіркою правильності**
- **оцінкою ймовірності використання ШІ**
- **5-бальною оцінкою з урахуванням −60% штрафу, якщо ШІ-ймовірність висока**
- **підрахунком середнього бала**

Ваші початкові відповіді оцінюються окремо для кожного пункту.

✅ 1. Аналіз відповідей (детально для кожного питання)

Питання 1

✔ Правильність:

Ви повністю правильно пояснили геометричний зміст формули належності до кола.
Жодних помилок.

✔ Повнота:

Враховано:

- центр кола,
- радіус,
- логіку порівняння з r^2.

Повна відповідь.

✔ Ймовірність використання ШІ:

Формулювання природне, коротке, без шаблонних конструкцій, характерних для ШІ.
Ймовірність: низька.

Оцінка: 5/5 (без штрафу)

Питання 2

✔ Правильність:

Повністю правильно наведені:

- зв’язок гамма-функції з факторіалом,
- причина використання `tgamma`,
- переваги при великих числах.

✔ Повнота:

Більш ніж достатня.

✔ ШІ-ймовірність:

Відповідь добре структурована, але стилістично нормальна для студента, який розуміє тему.

Ймовірність: середня.
Не достатньо висока для штрафу.

Оцінка: 5/5 (без штрафу)

Питання 3

✔ Правильність:

Суть збіжності та розбіжності описана вірно.
Але приклад параметрів eps і g **вийшов за межі дозволеного діапазону**, тому він некоректний для цієї задачі.

✔ Повнота:

Суть відповіді непогана, але приклад некоректний.

✔ ШІ-ймовірність:

Стиль відповіді короткий, не надто академічний —
Ймовірність: низька.

Оцінка: 4/5 (без штрафу)

Питання 4

✔ Правильність:

Причина застосування `while(true)` пояснена абсолютно вірно.

✔ Повнота:

Відповідь коротка, але повна. Більше нічого й не потрібно.

✔ ШІ-ймовірність:

Стиль “людський”, без типової ШІ-структурності.
Ймовірність: низька.

Оцінка: 5/5 (без штрафу)

Питання 5

✔ Правильність:

Чітко описано переваги:

- тестування,
- читабельність,
- розширюваність.

Повністю коректно.

✔ Повнота:

Повноцінна відповідь без недоліків.

✔ ШІ-ймовірність:

Стиль трохи шаблонний, але не явно ШІ-подібний.
Ймовірність: середня, але не висока.

Оцінка: 5/5 (без штрафу)

✅ 2. Підсумкова таблиця

№	Оцінка до штрафу	Ймовірність ШІ	Штраф	Кінцева оцінка

1	5	низька	0%	**5.0**
2	5	середня	0%	**5.0**
3	4	низька	0%	**4.0**
4	5	низька	0%	**5.0**
5	5	середня	0%	**5.0**

Штраф не застосовано, оскільки жодна відповідь не мала *високої* ймовірності бути згенерованою ШІ.

✅ 3. Загальна середня оцінка

$$
\frac{5 + 5 + 4 + 5 + 5}{5} = 4.8
$$

⭐ Підсумок: ваша загальна оцінка — 4.8 / 5

Чудові відповіді — логічні, точні, зрозумілі, з мінімальними зауваженнями.