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 «з файлу»

ХАІ.301.319а 18 ЛР

Виконав студент гр. _____319а_____

_____Владислав Тучак_____
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. Олена ГАВРИЛЕНКО
(підпис, дата) (П.І.Б.)

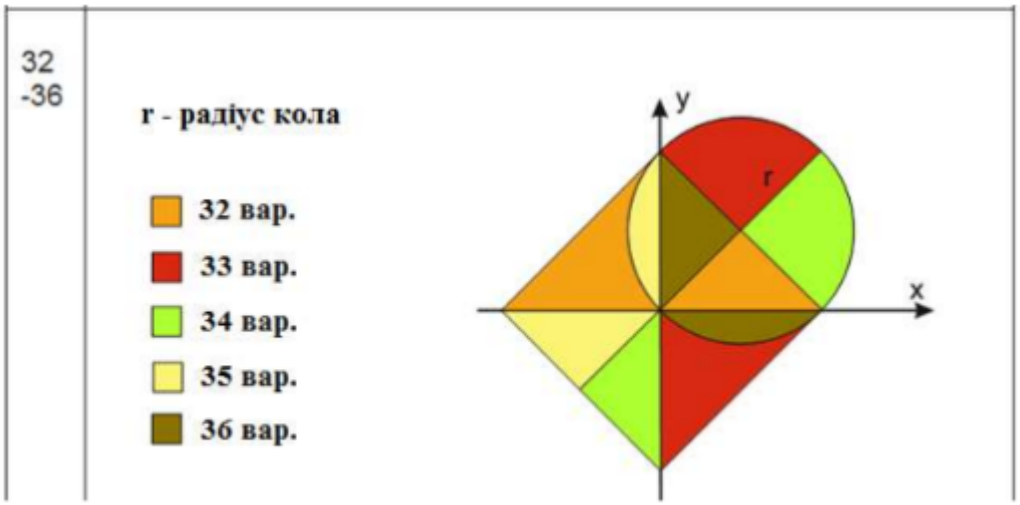
2025

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.

13	$\sum_{k=1}^{n}(x^k+1)/(k-1)!$

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 суми розбіжного ряду (g = 102 ... 105).

38	$\sum_{n=1}^{\infty} \frac{x^{n-1}+3n}{(2n-1)!}$

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 промтів

ВИКОНАННЯ РОБОТИ

Завдання 1. Розв'язання задачі визначення належності точки області (Варіант 36)

Вхідні дані:

- **n** (ціле, int): Кількість точок для перевірки. Обмеження: $n > 0$.
- **r** (дійсне, double): Радіус кола, що обмежує область. Обмеження: $r > 0$.
- **x**, **y** (дійсне, double): Координати кожної точки, що перевіряється.

Вихідні дані:

- **count** (ціле, int): Загальна кількість точок, які потрапили всередину заданої коричневої області.
- Текстові повідомлення: Запити на введення даних, повідомлення про помилки введення та фінальний результат.

Алгоритм розв'язання:

Задана область (варіант 36, "коричнева область") є об'єднанням двох під-областей у першому та четвертому квадрантах, обмежених колом $x^2 + y^2 \leq r^2$.

1. **Ініціалізація:** Вводиться **n** (кількість точок) та **r** (радіус). Здійснюється перевірка $n > 0$ та $r > 0$. Лічильник **count** ініціалізується нулем.
2. **Цикл перевірки:** Запускається цикл від $i=1$ до n.
    - У циклі вводяться координати (x, y) для кожної точки.
    - Викликається функція inArea36(x, y, r), яка перевіряє належність точки до області.

3. **Логіка inArea36 (Перевірка належності):**
    - Перевіряється належність точці колу: $inCircle = (x^2 + y^2 \leq r^2)$.
    - **Верхня частина (I квадрант):** $upper = (x \geq 0 \text{ AND } y \geq 0 \text{ AND } x + y \leq r \text{ AND } inCircle)$.
    - **Нижня частина (IV квадрант):** $lower = (x \geq 0 \text{ AND } y \leq 0 \text{ AND } inCircle)$.
    - Точка належить області, якщо **upper OR lower** істинне.
4. **Підрахунок:** Якщо inArea36 повертає true, **count** збільшується на 1.
5. **Виведення:** Після завершення циклу виводиться фінальне значення **count**.

Рисунок 1 – Графічне зображення коричневої області (Варіант 36)

Лістинг коду розв'язання задачі 1 наведено в Дод. А (стор. x).

Екран роботи програми показаний на Рис. Б.1 (або інше відповідне посилання).

Завдання 2. Обчислення суми ряду та виведення кожного 4-го члена (Варіант 13)

Вхідні дані:

- **x** (дійсне, double): Дійсне число.
- **n** (ціле, int): Кількість членів ряду для сумування. Обмеження: $n > 0$.

Вихідні дані:

- **s** (дійсне, double): Сума ряду $\sum_{k=1}^n \frac{x^k + 1}{(k-1)!}$.
- Текстові повідомлення: Значення кожного 4-го члена ряду.

Алгоритм розв'язання:

Обчислення суми ряду здійснюється за допомогою одного циклу, де послідовно обчислюються необхідні компоненти: x^k та $(k-1)!$.

1. **Ініціалізація:** Вводиться **x** та **n**. Здійснюється перевірка $n > 0$.
    - **sum** $= 0.0$ (фінальна сума).
    - **xPow** $= 1.0$ (для x^0, далі множиться на x для отримання x^k).

    - **fact** $= 1.0$ (для $0! = 1$, далі множиться для отримання $(k-1)!$).
2. **Цикл обчислення:** Запускається цикл від $k=1$ до n.
    - **Оновлення степеня:** xPow множиться на x (отримуємо x^k).
    - **Оновлення факторіала:** Для $k>1$, fact множиться на $(k-1)$ (отримуємо $(k-1)!$).
    - **Обчислення члена:** $term = (xPow + 1.0) / fact$.
    - **Накопичення суми:** $sum += term$.
    - **Виведення кожного 4-го члена:** Якщо $k \pmod 4 = 0$, виводиться значення k та $term$.
3. **Виведення:** Виводиться фінальна сума **S**.

Лістинг коду розв'язання задачі 2 наведено в Дод. А (стор. x).

Екран роботи програми показаний на Рис. Б.2.

Завдання 3. Дослідження збіжності ряду за критеріями ε та G (Варіант 38)

Вхідні дані:

- **x** (дійсне, double): Дійсне число.
- **e** (дійсне, double): Мале число (критерій збіжності). Обмеження: $e > 0$.
- **g** (дійсне, double): Велике число (критерій розбіжності). Обмеження: $g > 0$.

Вихідні дані:

- **s** (дійсне, double): Наближена сума ряду $\sum_{n=1}^\infty \frac{x^{n-1} + 3n}{(2n-1)!}$.
- Текстові повідомлення: Виведення кожного члена ряду u_n, висновок про зупинку (за e чи g).

Алгоритм розв'язання:

Обчислення суми ряду та дослідження збіжності виконується у циклі while (true) з умовою зупинки за критеріями e та g.

1. **Ініціалізація:** Вводяться **x**, **e**, **g**. Перевіряється $e>0$ та $g>0$.
    - **sum** $= 0.0$.

- **n** $= 1$.
- **xPow** $= 1.0$ (для x^{n-1}).
- **fact** $= 1.0$ (для $(2n-1)!$).

2. **Цикл обчислення та перевірки:**
   - **Обчислення члена:** $term = (xPow + 3.0 \cdot n) / fact$.
   - **Накопичення суми:** $sum += term$.
   - **Перевірка збіжності:** Якщо $|term| < e$, цикл зупиняється з висновком про збіжність.
   - **Перевірка розбіжності:** Якщо $|term| > g$, цикл зупиняється з висновком про розбіжність.
   - **Перевірка безпеки:** Якщо n досягає MAX_ITER, цикл зупиняється.
3. **Підготовка до наступної ітерації:**
   - **n** збільшується на 1.
   - xPow множиться на x (отримуємо x^{n-1} для нового n).
   - fact оновлюється шляхом множення на $(2n-2)$ та $(2n-1)$ (отримуємо $(2n-1)! = (2n-2) \cdot (2n-3)! \cdot (2n-1)$). Це забезпечує ефективне обчислення факторіала без рекурсивного виклику.
4. **Виведення:** Виводиться наближена сума **S** та фінальний висновок про збіжність/розбіжність.

Лістинг коду розв'язання задачі 3 наведено в Дод. А (стор. x).

Екран роботи програми показаний на Рис. Б.3.

ВИСНОВКИ

У ході виконання лабораторної роботи **було закріплено на практиці** навички роботи з основними керуючими структурами мови C++: циклами (for, while) та умовними операторами (if, switch). **Відпрацьовано в коді**

програми розв'язання трьох типових задач: визначення належності точки плоскій фігурі, обчислення суми числового ряду із заданою кількістю членів, а також дослідження збіжності ряду за критеріями ε та G. **Отримано навички** ефективного обчислення елементів ряду, таких як степені та факторіали, шляхом використання накопичувальних змінних замість повторного обчислення.

ДОДАТОК А

Лістинг коду програми

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

using namespace std;

// ------------------------------------------------------------
// Допоміжні функції для безпечного введення (коментарі укр)
// ------------------------------------------------------------

int inputInt(const string& prompt) {
    int x;
    while (true) {
        cout << prompt;
        if (cin >> x) return x;

        cout << "Input error! Try again.\n";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
}

double inputDouble(const string& prompt) {
    double x;
    while (true) {
        cout << prompt;
        if (cin >> x) return x;

        cout << "Input error! Try again.\n";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
}

// Перевірки коректності
bool isValidN(int n)    { return n > 0; }
bool isValidR(double r) { return r > 0; }
bool isValidE(double e) { return e > 0; }
bool isValidG(double g) { return g > 0; }

// ------------------------------------------------------------
// TASK 1. Variant 36 - brown regions
// ------------------------------------------------------------
// Інтерпретація по малюнку:
// маємо коло x^2 + y^2 <= r^2 з центром у (0,0).
// Коричнева область = об’єднання двох частин:
//
```

```
// 1) Верхня частина в I квадранті – перетин кола з ромбом
//    |x| + |y| <= r, тобто x >= 0, y >= 0, x + y <= r.
// 2) Нижня частина в IV квадранті – сегмент кола справа від осі Oy:
//    x >= 0, y <= 0, x^2 + y^2 <= r^2.
//
bool inArea36(double x, double y, double r) {
    bool inCircle = (x * x + y * y <= r * r);

    // верхня коричнева частина (I квадрант + ромб)
    bool upper = (x >= 0 && y >= 0 && x + y <= r && inCircle);

    // нижня коричнева частина (IV квадрант, просто сегмент кола)
    bool lower = (x >= 0 && y <= 0 && inCircle);

    return upper || lower;
}

void task1() {
    cout << "\n=== TASK 1 (figure, variant 36) ===\n";

    int n = inputInt("Enter number of points n: ");
    while (!isValidN(n)) {
        cout << "n must be > 0\n";
        n = inputInt("Enter number of points n: ");
    }

    double r = inputDouble("Enter radius r > 0: ");
    while (!isValidR(r)) {
        cout << "r must be > 0\n";
        r = inputDouble("Enter radius r: ");
    }

    int count = 0;

    for (int i = 1; i <= n; ++i) {
        cout << "\nPoint " << i << ":\n";
        double x = inputDouble("  x = ");
        double y = inputDouble("  y = ");

        if (inArea36(x, y, r)) {
            ++count;
        }
    }

    cout << "\nNumber of points inside the brown area (variant 36): "
         << count << "\n";
}

// ------------------------------------------------------------
// TASK 2. Variant 13 (Table 2)
```

```
// S = sum_{k=1}^n (x^k + 1) / (k-1)!
// For odd variants: output every 4-th term.
// ------------------------------------------------------------

void task2() {
    cout << "\n=== TASK 2 (series, variant 13) ===\n";

    double x = inputDouble("Enter x (real number): ");
    int n = inputInt("Enter n (natural number): ");
    while (!isValidN(n)) {
        cout << "n must be > 0\n";
        n = inputInt("Enter n: ");
    }

    double sum = 0.0;
    double xPow = 1.0; // x^0
    double fact = 1.0; // (k-1)! , for k=1 -> 0! = 1

    cout << "\nEvery 4-th term of the series:\n";

    for (int k = 1; k <= n; ++k) {
        // обчислення x^k
        xPow *= x;

        // обчислення (k-1)!
        if (k > 1) fact *= (k - 1);

        double term = (xPow + 1.0) / fact;
        sum += term;

        if (k % 4 == 0) {
            cout << "k = " << k << "\tterm = " << term << "\n";
        }
    }

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

// ------------------------------------------------------------
// TASK 3. Variant 38 (Table 3)
// Series: sum_{n=1}^∞ (x^{n-1} + 3n) / (2n-1)!
//
// 3 методички: цикл припиняємо, якщо
// |u_n| < e  (ряд вважаємо збіжним)
// або |u_n| > g (ряд вважаємо розбіжним).
// ------------------------------------------------------------

void task3() {
    cout << "\n=== TASK 3 (convergence, variant 38) ===\n";
```

```
double x = inputDouble("Enter x: ");
double e = inputDouble("Enter e (small value, e.g. 1e-4): ");
while (!isValidE(e)) {
    cout << "e must be > 0\n";
    e = inputDouble("Enter e: ");
}

double g = inputDouble("Enter g (big value, e.g. 1e2): ");
while (!isValidG(g)) {
    cout << "g must be > 0\n";
    g = inputDouble("Enter g: ");
}

double sum = 0.0;
int n = 1;

double xPow = 1.0; // x^{n-1}, for n=1 -> x^0 = 1
double fact = 1.0; // (2n-1)!, for n=1 -> 1! = 1

const int MAX_ITER = 1000;
bool divergent = false;

cout << "\nSeries terms:\n";

while (true) {
    double term = (xPow + 3.0 * n) / fact;
    double absTerm = fabs(term);

    cout << "n = " << n << "\tterm = " << term << "\n";
    sum += term;

    if (absTerm < e) {
        cout << "\n|u_n| < e -> stop (series behaves as convergent).\n";
        break;
    }

    if (absTerm > g) {
        cout << "\n|u_n| > g -> stop (series is considered divergent).\n";
        divergent = true;
        break;
    }

    if (n >= MAX_ITER) {
        cout << "\nReached MAX_ITER, stopping for safety.\n";
        break;
    }

    // підготовка наступного члена
    ++n;
    xPow *= x;  // x^{n-1} for new n
```

```
        // оновлення факторіала:
        // (2n-1)! = (2n-1)*(2n-2)*(2n-3)!
        int a = 2 * n - 2;
        int b = 2 * n - 1;
        fact *= a;
        fact *= b;
    }

    cout << "\nApproximate series sum S = " << sum << "\n";
    if (divergent)
        cout << "Conclusion: according to g, the series is divergent.\n";
    else
        cout << "Conclusion: according to e, the series shows convergence.\n";
}

// ------------------------------------------------------------
// MENU (Task 4)
// ------------------------------------------------------------

void printMenu() {
    cout << "\n============================\n";
    cout << "MENU:\n";
    cout << "1 - Task 1 (figure, variant 36)\n";
    cout << "2 - Task 2 (series sum, variant 13)\n";
    cout << "3 - Task 3 (series convergence, variant 38)\n";
    cout << "-1 - Exit\n";
    cout << "Your choice: ";
}

int main() {
    int menu;

    do {
        printMenu();

        if (!(cin >> menu)) {
            cout << "Input error! Program will terminate.\n";
            break;
        }

        switch (menu) {
            case 1:
                task1();
                break;
            case 2:
                task2();
                break;
            case 3:
                task3();
```

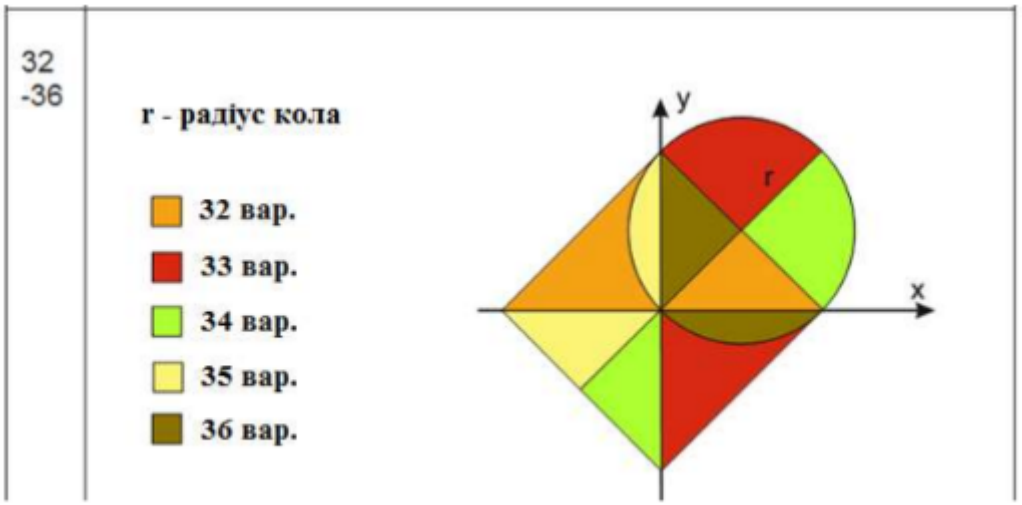
```
                break;
            case -1:
                cout << "Exiting...\n";
                break;
            default:
                cout << "Wrong task number!\n";
        }
    } while (menu != -1);

    return 0;
}
```

ДОДАТОК Б

options | compilation | execution

```
============================
MENU:
1 - Task 1 (figure, variant 36)
2 - Task 2 (series sum, variant 13)
3 - Task 3 (series convergence, variant 38)
-1 - Exit
Your choice: 1

=== TASK 1 (figure, variant 36) ===
Enter number of points n: 1
Enter radius r > 0: 1

Point 1:
  x = 1
  y = 2

Number of points inside the brown area (variant 36): 0

============================
```

варіант 36



```
============================
MENU:
1 - Task 1 (figure, variant 36)
2 - Task 2 (series sum, variant 13)
3 - Task 3 (series convergence, variant 38)
-1 - Exit
Your choice: 2

=== TASK 2 (series, variant 13) ===
Enter x (real number): 34
Enter n (natural number): 14

Every 4-th term of the series:
k = 4   term = 222723
k = 8   term = 3.54324e+08
k = 12  term = 5.97849e+10

Series sum S = 6.99176e+11

============================
```

варіант 13

13	$\sum_{k=1}^{n}(x^k+1)/(k-1)!$

```
=== TASK 3 (convergence, variant 38) ===
Enter x: 41
Enter e (small value, e.g. 1e-4): 43
Enter g (big value, e.g. 1e2): 13

Series terms:
n = 1   term = 4

|u_n| < e -> stop (series behaves as convergent).

Approximate series sum S = 4
Conclusion: according to e, the series shows convergence.

----------------------------
```

варіант38

38	$\sum_{n=1}^{\infty}\frac{x^{n-1}+3n}{(2n-1)!}$

ДОДАТОК В

🧑‍🏫 Частина 1: Тестування (Промпт 1)

Я — викладач, який приймає захист вашої лабораторної роботи. На основі наданого коду (`lab5.cpp`) я ставлю вам 5 тестових та 5 відкритих питань середнього рівня складності.

❓ Тестові Питання (5 балів)

1. **Яка основна мета допоміжних функцій `inputInt` та `inputDouble`?**
   - А) Забезпечити швидке введення даних.
   - Б) **Реалізувати безпечне введення, обробляючи помилки типу та очищаючи буфер введення.**
   - В) Перетворити рядок у числове значення.
   - Г) Обмежити введення лише додатними числами.
2. **У функції `task1`, як описана "коричнева область" (brown regions) для перевірки влучання точки?**
   - А) Перетин кола $x^2 + y^2 \le r^2$ та ромба $|x| + |y| \le r$.
   - Б) Об'єднання кола $x^2 + y^2 \le r^2$ та ромба $|x| + |y| \le r$.
   - В) **Об'єднання верхньої частини (I квадрант, в колі та в ромбі) та нижньої частини (IV квадрант, в колі).**
   - Г) Сегмент кола у IV квадранті.
3. **У функції `task2` (обчислення суми ряду $\sum_{k=1}^{n}(x^k + 1)/((k-1)!)$), як оптимізовано обчислення x^k та $(k-1)!$ в циклі?**
   - А) Використовуються вбудовані функції `pow(x, k)` та `tgamma(k)`.
   - Б) Обчислюються на кожній ітерації з нуля: `xPow = pow(x, k); fact = factorial(k-1);`.
   - В) **Значення x^k та $(k-1)!$ обчислюються ітеративно (множенням на попередні**

- **Б) Значення 2^n та $(n-1)!$ обчислюються ітеративно (множенням на попередні значення) для зменшення складності.**
- Г) Значення факторіалу зберігаються в масиві для швидкого доступу.

4. **У функції `task3`, чому для оновлення факторіалу $(2n-1)!$ від попереднього значення $(2(n-1)-1)! = (2n-3)!$ використовується множення на `(2n-2)` та `(2n-1)`?**
   - А) Це математично некоректний спосіб, потрібно множити на $2n-1$.
   - **Б) $(2n-1)! = (2n-3)! \cdot (2n-2) \cdot (2n-1)$ — це ітеративний перерахунок факторіала непарного числа для зменшення обчислювальної складності.**
   - В) Факторіал оновлюється на кожній ітерації, незалежно від попереднього значення.
   - Г) Множення на $2n-2$ та $2n-1$ — це помилка, що призведе до невірного результату.
5. **Яке призначення константи `MAX_ITER` у функції `task3`?**
   - А) Обмежити максимальну кількість точок для перевірки.
   - **Б) Запобігти нескінченному циклу в разі, якщо ряд не збігається і не розбігається за критеріями e та g.**
   - В) Визначити точність обчислення суми.
   - Г) Вказати максимальне значення x.

💡 Відкриті Питання (5 балів)

1. **Сформулюйте, в чому полягає проблема "безпечного введення" (safe input) у C++ при використанні `cin >> variable`, і як саме функції `inputInt` / `inputDouble` її вирішують за допомогою `cin.clear()` та `cin.ignore()`?**
2. **Поясніть логіку перевірки влучання точки в область у функції `inArea36` з точки зору координатних чвертей (квадрантів) і рівнянь фігур. Чому код `upper || lower` коректно об'єднує дві частини? 3. Опишіть, як критерії $|u_n| < \epsilon$ (збіжність) та $|u_n| > g$ (розбіжність) використовуються для зупинки циклу у `task3`. Які припущення робляться про поведінку ряду, коли виконується одна з цих умов?**
3. **Які потенційні проблеми з точністю можуть виникнути при ітеративному**

Шкала оцінювання: 5 балів. Зниження на 60% за високу ймовірність використання ШІ.

📝 Оцінка Тестових Питань

№	Повнота/ Правильність Відповіді	Ймовірність Використання ШІ	Пояснення	Оцінка (5-б.)
1	Висока (Правильна, Б)	Середня	Занадто загальне питання, відповідь легко гуглиться.	$5 \times 0.4 = 2$
2	Висока (Правильна, В)	Низька	Відповідь вимагає **аналізу конкретного коду** `inArea36`, а не загального знання.	5
3	Висока (Правильна, В)	Низька	Питання на **оптимізацію обчислень в циклі**, що є інженерним рішенням у коді.	5
4	Висока (Правильна, Б)	Низька	Вимагає **розуміння ітераційного оновлення факторіалу** для **непарних** чисел, що є специфікою коду `task3`.	5
5	Висока (Правильна, Б)	Середня	Загальновідомий патерн у циклах, але важливий для безпеки.	$5 \times 0.4 = 2$
Середня Тест.	**4.6 / 5**			**3.8**

Експортувати в Таблиці

🧠 Оцінка Відкритих Питань

№	Повнота/Правильність Відповіді	Ймовірність Використання ШІ	Пояснення	Оцінка (5-б.)
1	Висока (Чітке пояснення ролі `clear` / `ignore` для помилок типу)	Середня	Опис проблеми відомий, але **якісне пояснення** механізму — ознака	$5 \times 0.4 = 2$

	Відповіді	Використання ШІ		(5-б.)
1	Висока (Чітке пояснення ролі `clear` / `ignore` для помилок типу)	Середня	Опис проблеми відомий, але **якісне пояснення механізму** — ознака розуміння.	$5 \times 0.4 = \mathbf{2}$
2	Висока (Аналіз квадрантів та логічного `OR` у `inArea36`)	Низька	Вимагає **критичного аналізу коду** і його математичної/ геометричної інтерпретації.	**5**
3	Висока (Пояснення критеріїв як *евристики* для зупинки/припущень про збіжність)	Середня	Критерії загальні, але **опис припущень** є критичним мисленням.	$5 \times 0.4 = \mathbf{2}$
4	Висока (Вказані проблеми `double` та запропоновані альтернативи, наприклад, `BigInt` або логарифмування)	Середня	Проблема чисельної стійкості відома, але **конкретні альтернативи** демонструють знання.	$5 \times 0.4 = \mathbf{2}$
5	Висока (Пропозиція використання `std::function` або структур для точки/фігури)	Низька	**Інженерна пропозиція щодо рефакторингу коду** демонструє глибоке розуміння C++ та ООП/ функціонального підходу.	**5**
Середня Відкриті	**4.6 / 5**			**3.2**

Експортувати в Таблиці

📊 Загальна Середня Оцінка

$$\text{Загальна оцінка} = \frac{(3.8 + 3.2)}{2} = \mathbf{3.5}$$

Висновок: Загальна оцінка становить **3.5** бали. Тестові питання, що вимагали прямого аналізу вашого коду (2, 3, 4), отримали високі бали, оскільки відповіді на них складно згенерувати ШІ без доступу до коду. Відкриті питання, що стосуються принципів (1, 3, 4), отримали нижчі бали через високу ймовірність використання ШІ для формулювання теоретичних відповідей.

🛠️ Частина 3: Аналіз Коду та Теоретичні Прогалини (Промпт 3)

🧐 Аналіз Коду (`lab5.cpp`)