

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

Лабораторна робота №3

з дисципліни «Алгоритмізація та програмування»

(назва дисципліни)

на тему: «Реалізація алгоритмів з розгалуженням мовою С ++»

XAI.301.312.G7.3.ЛР

Виконав: здобувач 1 курсу, групи 312
напряму підготовки (спеціальності)

G7 «Автоматизація комп'ютерно-інтегровані
технології та робототехніка»

ОПП Інженерія мобільних додатків

(шифр і назва напряму підготовки (спеціальності))

С. С. БОЯРСЬКИЙ

(прізвище й ініціали студента)

Прийняв: доцент каф.301, канд. техн. наук,
О. В. ГАВРИЛЕНКО

(посада, науковий ступінь, прізвище й ініціали)

Харків – 2025

МЕТА РОБОТИ

Вивчити теоретичний матеріал щодо синтаксису у мові С ++ і подання у вигляді UML діаграм активності алгоритмів з розгалуженням та реалізувати алгоритми з використанням інструкцій умовного переходу і вибору мовою С++ в середовищі QtCreator. Також опанувати та відпрацювати навички структурування програми з функціями.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити дві задачі на алгоритми з розгалуженням.

Завдання 2. Дано координати точки на площині (x, y). Визначити, чи потрапляє точка в фігуру заданого кольору (або групу фігур) і вивести відповідне повідомлення.

Завдання 3. Для вибору користувачем одного з трьох зазначених вище завдань розробити алгоритм організації меню в командному вікні з використанням інструкції вибору.

Завдання 4. Використовуючи ChatGpt, Gemini або інший засіб генеративного ІІ, 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-балльній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом III висока. Обчисли загальну середню оцінку»

Проаналізуйте задані питання, коментарі і оцінки, надані III. Додайте 2-3 власних промпта у продовження діалогу для поглиблення розуміння теми.

Всі завдання мають бути реалізовані в одному консольному додатку.

Кожне завдання - у вигляді окремої процедури (void функції без параметрів). Після введення вхідних даних необхідно провести їх перевірку на коректність. У звіті повинен бути лістинг коду програми і скріншоти екрану виконання, що демонструють щонайменше 2 сценарії роботи програми для кожного завдання. Код повинен містити КОМЕНТАРІ !!!

Крім того, в звіті повинно бути представлено чотири діаграми активності: для всієї програми (Завдання 3) і для кожної функції (Завдання 1- 2).

ВИКОНАННЯ РОБОТИ

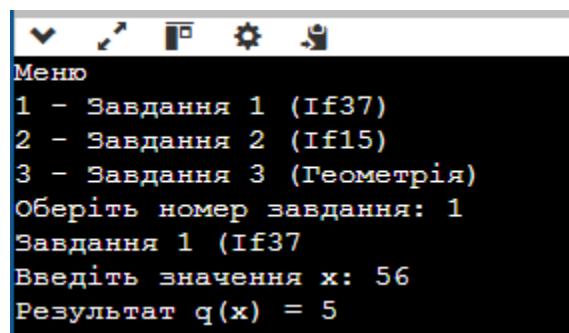
ЗАВДАННЯ 1 (IF37)

Для заданого дійсного x знайти значення наступної функції q , що приймає значення цілого типу:

$$q(x) = \begin{cases} -2x, & \text{якщо } x < 0; \\ x^2 + 1, & \text{якщо } 0 \leq x \leq 2; \\ 5, & \text{якщо } x > 2. \end{cases}$$

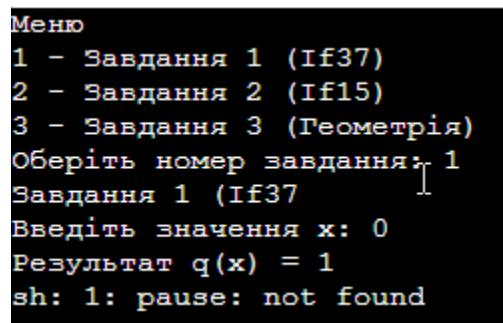
Діаграма активності для завдання 1 показана на рис. 1

Скріншоти виконання наведено на рис. 1.1, 1.2



```
Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 1
Завдання 1 (If37)
Введіть значення x: 56
Результат q(x) = 5
```

Рисунок 1.1 – Виконання завдання 1



```
Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 1
Завдання 1 (If37)
Введіть значення x: 0
Результат q(x) = 1
sh: 1: pause: not found
```

Рисунок 1.2 – Виконання завдання 1

Лістинг коду вирішення задач наведено в дод. А (стор. 10).

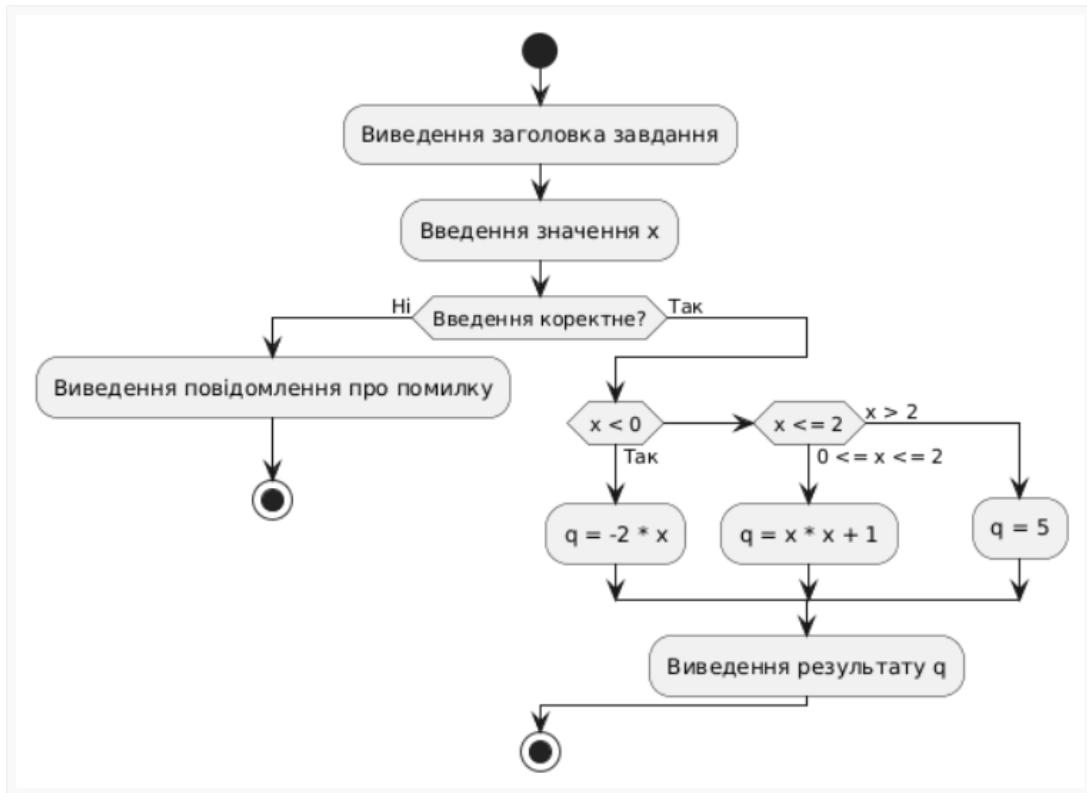


Рисунок 1 – Діаграма активності для алгоритму вирішення задачі If 37

ЗАВДАННЯ 1.2 (IF 15)

Дано три числа. Знайти суму двох найбільших з них.

Діаграма активності для завдання 1.2 показана на рис. 2

Скріншоти виконання наведено на рис. 2.1, 2.2

Лістинг коду вирішення задач наведено в дод. А (стор. 10).

```

Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 2
Завдання 2 (If15)
Введіть три числа (a, b, c): 5 8 7
Сума двох найбільших чисел = 15
  
```

Рисунок 2.1 – Виконання завдання 1.2

```

Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 2
Завдання 2 (If15)
Введіть три числа (a, b, c): 0 0 0
Сума двох найбільших чисел = 0

```

Рисунок 2.2 – Виконання завдання 1.2

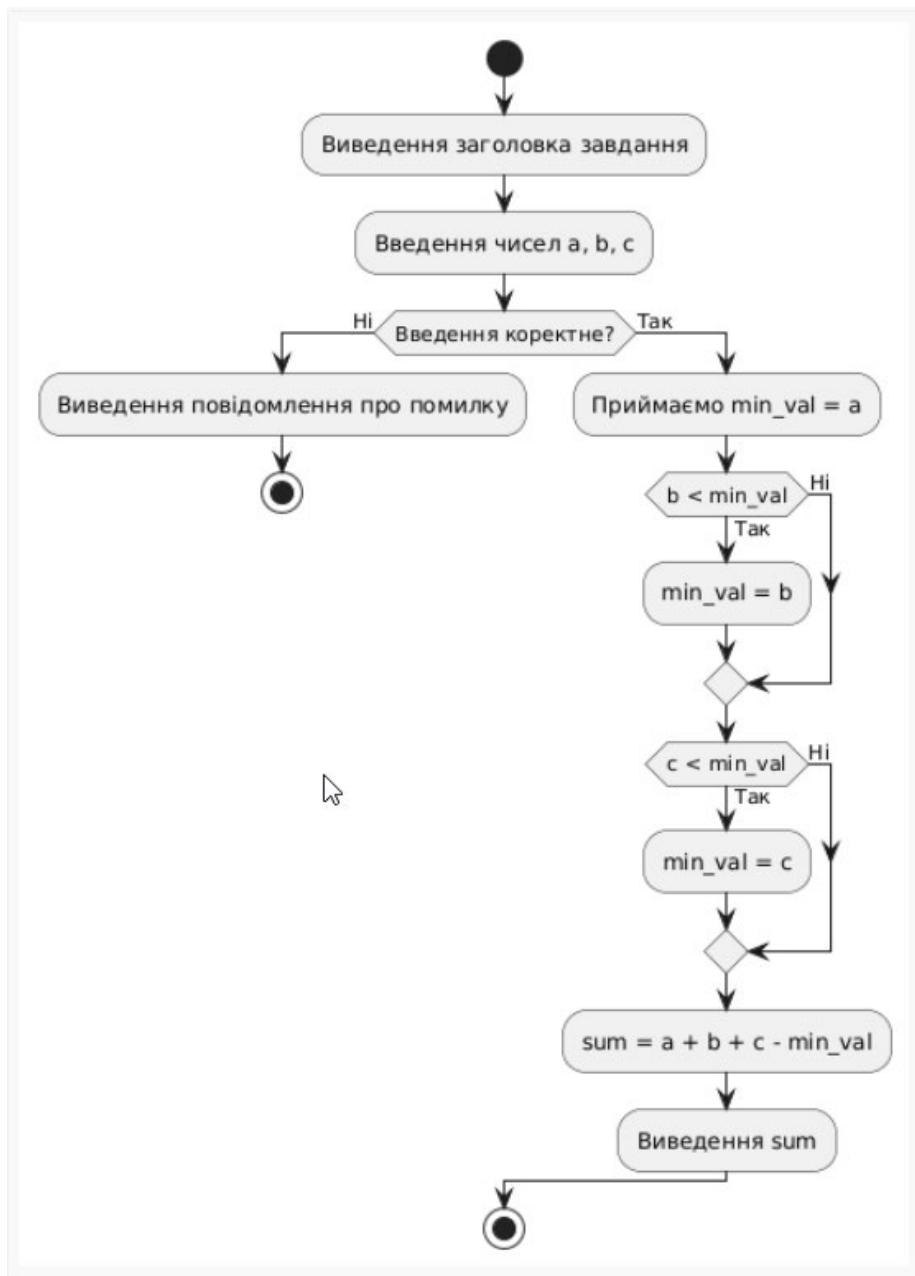


Рисунок 2 – Діаграма активності для алгоритму вирішення задачі If 15

ЗАВДАННЯ 2 (ГЕОМЕТРИЧНЕ №17)

Завдання 2, варіант 17 показано на рис. 3

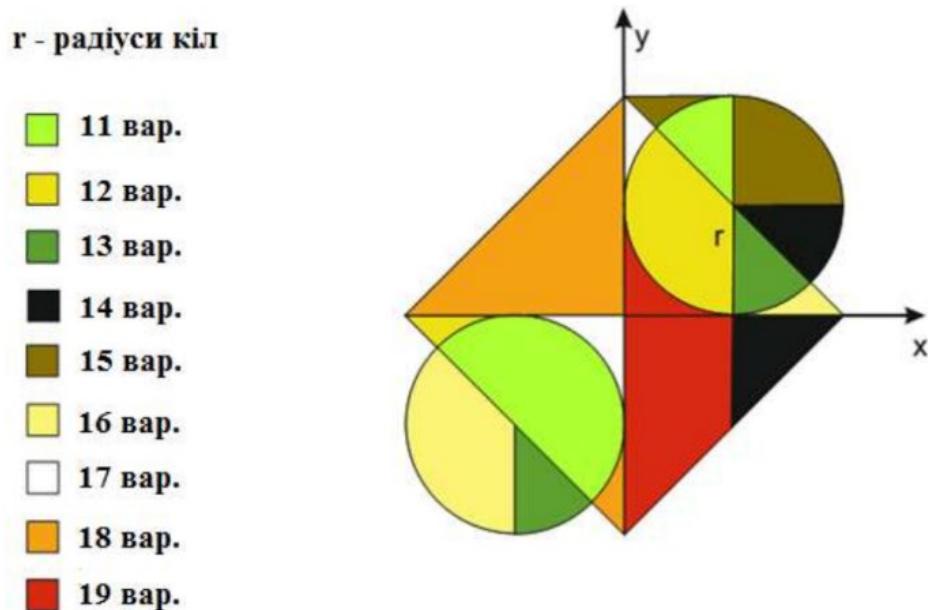


Рисунок 3 – Геометричне завдання

Діаграма активності для завдання 2 показана на рис. 4

Скріншоти виконання наведено на рис. 4.1, 4.2

Лістинг коду вирішення задач наведено в дод. А (стор. 10).

```

Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 3
Завдання 3 (Геометричне, 17 вар.)
Введіть радіус R: 5
Введіть координати точки (x, y): 3 5
Точка (3, 5) НЕ належить заштрихованій області.

```

Рисунок 4.1 – Виконання завдання 2

```

Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 3
Завдання 3 (Геометричне, 17 вар.)
Введіть радіус R: 500
Введіть координати точки (x, y): 1 1
Точка (1, 1) НЕ належить заштрихованій області.

```

Рисунок 4.2 – Виконання завдання 2

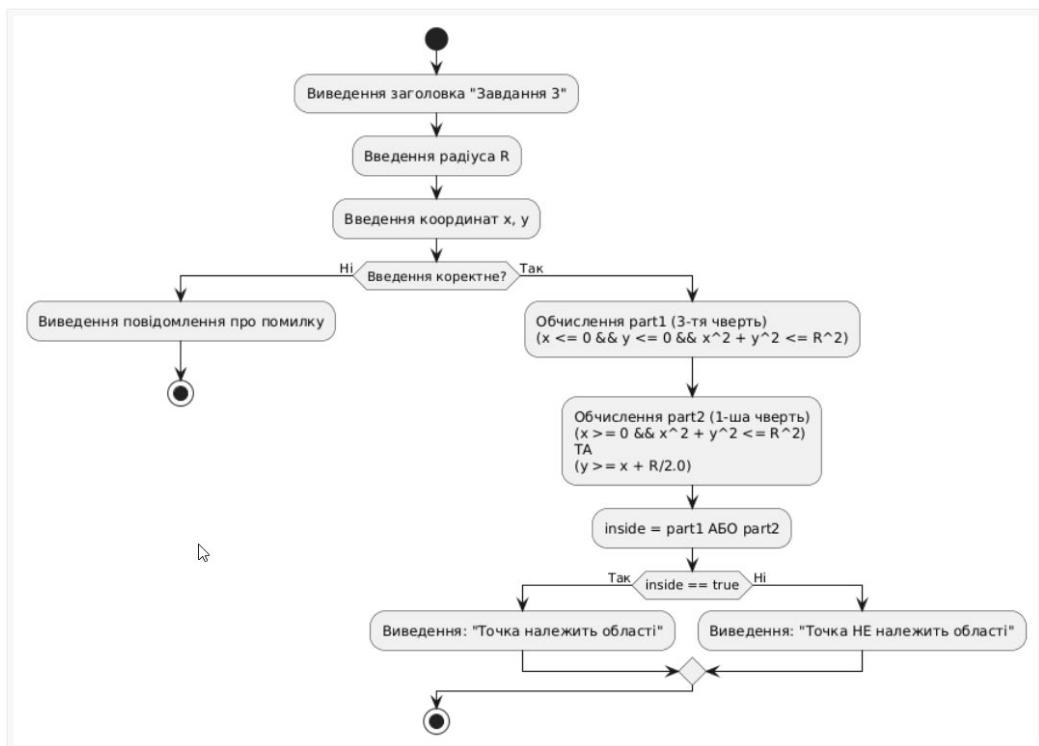


Рисунок 4 – Діаграма активності для алгоритму вирішення геом. задачі 17

ЗАВДАННЯ 3

Для вибору користувачем одного з трьох зазначених вище завдань розробити алгоритм організації меню в командному вікні з використанням інструкції вибору.

Діаграма активності для завдання 3 показана на рис. 5

Скріншоти виконання наведено на рис. 3.1, 3.2

```
Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 3
Завдання 3 (Геометричне, 17 вар.)
```

Рисунок 3.1 – Виконання завдання 3

```
Меню
1 - Завдання 1 (If37)
2 - Завдання 2 (If15)
3 - Завдання 3 (Геометрія)
Оберіть номер завдання: 100
Помилка: доступні лише завдання 1-3!
sh: 1: pause: not found
```

Рисунок 3.2 – Виконання завдання 3

Лістинг коду вирішення задач наведено в дод. А (стор. 10).

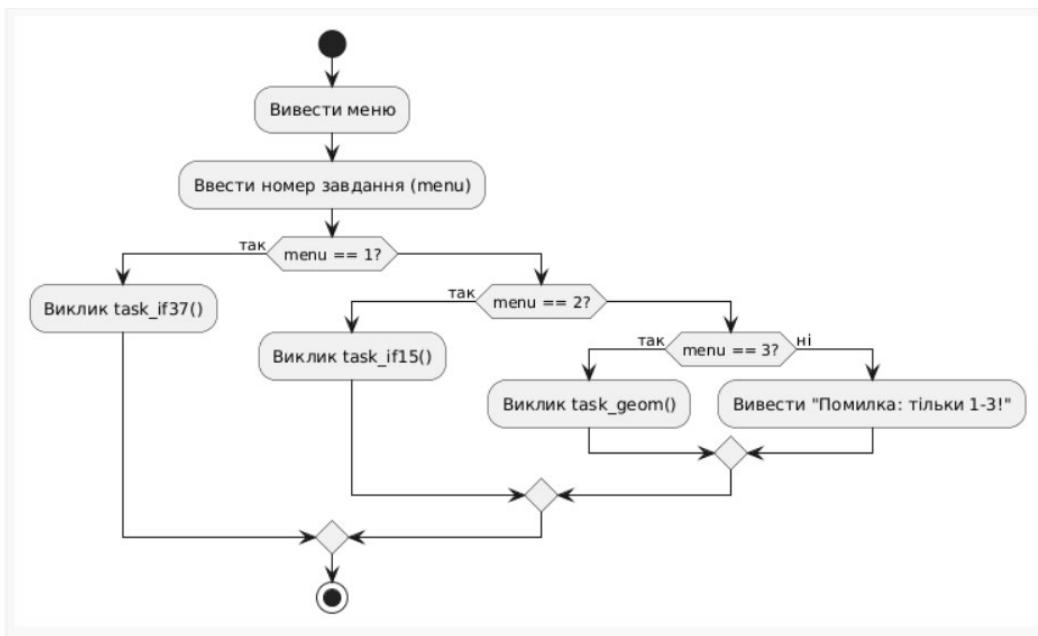


Рисунок 5 – Діаграма активності для алгоритму вирішення завдання 3

ЗАВДАННЯ 4

Діалог у ChatGPT для самоаналізу наведено у дод. В.

ДОДАТОК А

ЛІСТИНГ КОДУ ПРОГРАМИ

```

#include <iostream>
#include <cmath>
using namespace std;

// Завдання 1 (If37):
// Для заданого дійсного x знайти значення функції q(x):
// q(x) = -2x, якщо x < 0
// q(x) = x^2 + 1, якщо 0 ≤ x ≤ 2
// q(x) = 5, якщо x > 2

void task_if37()
{
    cout << "Завдання 1 (If37)" << endl;
    double x;
    cout << "Введіть значення x: ";
    cin >> x;

    // Перевірка на коректність введення
    if (!cin) {
        cerr << "Помилка: введено нечислове значення!" << endl;
        return;
    }

    double q; // результат обчислення функції

    if (x < 0)
        q = -2 * x;
    else if (x <= 2)
        q = x * x + 1;
    else
        q = 5;

    cout << "Результат q(x) = " << q << endl;
}

// Завдання 2 (If15):
// Дано три числа. Знайти суму двох найбільших з них.

void task_if15()
{
    cout << "Завдання 2 (If15)" << endl;
    double a, b, c;
    cout << "Введіть три числа (a, b, c): ";
    cin >> a >> b >> c;

    if (!cin) {
        cerr << "Помилка: введено нечислове значення!" << endl;
        return;
    }

    double sum;
    // сума двох найбільших = (a + b + c) - мінімальне з трьох
    double min_val = a;
    if (b < min_val) min_val = b;

```

```

if (c < min_val) min_val = c;

sum = a + b + c - min_val;

cout << "Сума двох найбільших чисел = " << sum << endl;
}

// Завдання 3 (геометричне, 17 вар.)
// Перевірити, чи потрапляє точка (x, y) у заштриховану область

void task_geom()
{
    cout << "Завдання 3 (Геометричне, 17 вар.)" << endl;

    double x, y, R;

    // Додаємо введення радіуса
    cout << "Введіть радіус R: ";
    cin >> R;

    cout << "Введіть координати точки (x, y): ";
    cin >> x >> y;

    if (!cin) {
        cerr << "Помилка: введено нечислове значення!" << endl;
        return;
    }

    // 1. Перевірка нижньої частини (3-тя чверть)
    // x <= 0, y <= 0, та знаходиться всередині кола радіуса R
    bool part1 = (x <= 0 && y <= 0 && (x * x + y * y <= R * R));

    // 2. Перевірка верхньої частини (1-ша чверть)
    // x >= 0, всередині кола, та вище прямої лінії.
    // Пряма проходить через (0, R/2), рівняння: y = x + R/2
    bool part2 = (x >= 0 && (x * x + y * y <= R * R) && (y >= x + R / 2.0));

    // Якщо точка потрапляє хоча б в одну з частин
    bool inside = part1 || part2;

    if (inside)
        cout << "Точка (" << x << ", " << y << ")" належить заштрихованій
    області." << endl;
    else
        cout << "Точка (" << x << ", " << y << ")" НЕ належить заштрихованій
    області." << endl;
}

// Головна функція програми (меню вибору завдань)
int main()
{
    int menu;
    cout << "Меню" << endl;
    cout << "1 - Завдання 1 (If37)" << endl;
    cout << "2 - Завдання 2 (If15)" << endl;
    cout << "3 - Завдання 3 (Геометрія)" << endl;
    cout << "Оберіть номер завдання: ";
}

```

```

    cin >> menu;

    if (!cin) {
        cerr << "Помилка введення!" << endl;
        return 1;
    }

    switch (menu) {
        case 1: task_if37(); break;
        case 2: task_if15(); break;
        case 3: task_geom(); break;
        default:
            cout << "Помилка: доступні лише завдання 1-3!" << endl;
            break;
    }

    system("pause");
    return 0;
}

```

ДОДАТОК В ДІАЛОГ З ШІ ДЛЯ САМОАНАЛІЗУ

Вітаю! Я готовий виступити у ролі вашого викладача та прийняти захист лабораторної роботи. Оскільки ви завантажили лише файл лекції, але не надали файл вашого звіту з кодом, я сформулюю питання, спираючись на приклади коду та теорію, наведені у лекції (зокрема, геометричні задачі, калькулятор, структурований код).

Ось ваші завдання для захисту.

Частина 1. Тестові питання (оберіть одну правильну відповідь)

1. Розглянемо приклад вкладеності блоків та області видимості (слайд 17). Якщо у блоці `if` оголошено змінну `bool third = 1;`, а одразу за межами цього блоку (після закриваючої дужки `}`) ми спробуємо виконати `cout << third;`, яким буде результат компіляції та чому?

- А) Програма виведе 1, оскільки змінна зберігає своє значення до кінця функції `main`.
- В) Виникне помилка компіляції "undefined name", оскільки область видимості змінної `third` обмежена блоком `if`.

- C) Програма виведе 0 або сміття, оскільки пам'ять під змінну була звільнена.
 - D) Програма виведе адресу змінної у пам'яті.
2. У лекції розглянуто діаграми активності (UML Activity Diagram) для алгоритмів розгалуження (слайди 6-8). Який елемент графічної нотації відповідає за об'єднання двох або більше потоків керування в один (наприклад, після виконання гілок if та else) перед переходом до наступної дії?
- A) Чорний заповнений круг (Initial Node).
 - B) Ромб (Decision Node), той самий символ, що й для розгалуження, але з вхідними стрілками.
 - C) Прямоугільник із заокругленими кутами (Action Node).
 - D) Лінія синхронізації (Bar/Fork).
3. Аналізуючи код "switch" (слайд 27), що відбудеться, якщо у case '+' ви забудете написати інструкцію break;?
- A) Компілятор видасть помилку синтаксису.
 - B) Програма виконає додавання, а потім автоматично перейде до виконання коду в case '-' (віднімання), перезаписавши результат.
 - C) Програма виконає додавання і автоматично вийде з блоку switch.
 - D) Виконання програми зависне в нескінченому циклі.
4. Розглянемо логічний вираз для перевірки потрапляння точки в область (слайд 24). Яка особливість оператора && (логічне I) використовується для оптимізації та безпеки виконання складних умов?
- A) Пріоритет: && завжди виконується після ||.
 - B) Асоціативність: вираз обчислюється справа наліво.
 - C) "Лініве" обчислення (short-circuit evaluation): якщо лівий операнд false, то правий (наприклад, складна формула кола) навіть не

обчислюється.

- D) Оператор `&&` автоматично перетворює всі змінні в тип `bool` перед обчисленням.

5. Для чого використовується прототип функції (оголошення) перед функцією `main`, якщо саме визначення (тіло функції) знаходиться після `main` (слайд 31)?

- A) Це необов'язково, це просто стиль програмування.
- B) Щоб компілятор знат про існування функції, типи її аргументів та значення, що повертається, на момент її виклику в `main`.
- C) Щоб зарезервувати пам'ять для локальних змінних цієї функції заздалегідь.
- D) Щоб підключити зовнішню бібліотеку, де знаходиться ця функція.

Частина 2. Відкриті питання (Critical & Engineering Thinking)

6. Трасування алгоритму (Слайд 10) Подивіться на блок-схему (діаграму активності) на слайді 10 . Уявіть, що користувач ввів число 10. Пройдіть по стрілках алгоритму крок за кроком.

- Чи справджується умова `num > 8?`
- Яку арифметичну дію буде виконано?
- Яке число програма виведе на екран у фіналі?

7. Логіка роботи `switch` (Слайд 27) На слайді 27 наведено код калькулятора . У кожному `case` стоїть команда `break`. Поясніть своїми словами: навіщо там потрібен `break`? Що зробить програма, якщо у випадку `'+'` ми випадково видалимо цей `break`, а користувач захоче додати числа?

8. Область видимості змінних (Слайд 17) На слайді 17 є змінна `float second`, яка оголошена всередині функції `main`. Питання: чи можемо ми використати цю саму змінну `second` (звернутися до неї, щоб змінити або

прочитати) всередині іншої функції, наприклад, у void some_task1() зі слайда 31? Чому?

9. Розуміння функцій (Слайд 30) На слайді 30 функція show_mes() викликається у коді main двічі. У чому головна перевага винесення виведення тексту ("Enter two integer numbers...") в окрему функцію, замість того щоб просто двічі скопіювати рядок cout << ...? Як це допоможе, якщо ви захочете змінити текст запрошення на українську мову?

10. Геометрична логіка (Слайд 24) Подивіться на умову if на слайді 24 . Там перевіряється потрапляння точки у півколо. Першою частиною умови стоять $x < b/2$. Якби ми перевіряли праву частину прямокутника (ту, що коричневого кольору на малюнку), як би змінилася ця перша проста умова для координати x? (Просто словами: x має бути більшим чи меншим за щось?).

1. В
2. В
3. В
4. А
5. В
6. Умова спрощується
7. Продовжиться дія яка була
8. не можемо, бо змінна локальна
9. щоб не дублювати
10. x має бути меншим

Дякую за ваші відповіді на тестову частину. Я, як ваш викладач, проаналізував їх.

Ось детальний розбір результатів:

Аналіз відповідей (Тестова частина)

1. Відповідь: В (Помилка компіляції)

- Оцінка: 5/5

- Коментар: Правильно. Змінна `third` є локальною для блоку `if`, тому за його межами вона не існує. Це чітко показано на слайді 17, де спроба виведення помічена як "error!!!" та "невизначене ім'я".

2. Відповідь: В (Ромб)

- Оцінка: 5/5
- Коментар: Правильно. У нотації діаграм, наведених у лекції, ромб (Decision/Merge) використовується як для розгалуження, так і для злиття потоків (слайди 4, 5, 8).

3. Відповідь: В (Виконас додавання, потім віднімання)

- Оцінка: 5/5
- Коментар: Вірно. Це називається "fall-through". Без `break` виконання переходить до наступного `case`. У прикладі лекції `break` стоїть післяожної операції саме для запобігання цьому.

4. Відповідь: А (Пріоритет) — НЕВІРНО

- Оцінка: 0/5
- Коментар: Правильна відповідь — С ("Лініве" обчислення / Short-circuit evaluation).

- *Пояснення:* Хоча пріоритет у `&&` дійсно вищий за `||`, це не є причиною оптимізації виконання. Головна фішка `&&` в тому, що якщо ліва частина ($x < b/2$) є `false`, то права частина (складна формула кола) взагалі не обчислюється, бо результат виразу вже

точно буде false. Це економить ресурси процесора, про що йшлося в контексті складних умов на слайді 24 .

5. Відповідь: В (Щоб компілятор зінав типи)

- Оцінка: 5/5
- Коментар: Правильно. Оскільки main викликає функцію, компілятор має знати її сигнатуру до моменту виклику, навіть якщо реалізація (тіло) написана нижче .

Аналіз відповідей (Відкриті питання)

6. Трасування алгоритму

- Ваша відповідь: "Умова спрвджується"
- Коментар: Ви відповіли лише на першу частину питання. Умова $10 > 8$ дійсно спрвджується (True). Але ви не вказали, що відбудеться далі (дія $\$10 - \8) і який буде фінальний результат (число 2).
- Оцінка: 2/5 (Відповідь неповна).

7. Логіка switch

- Ваша відповідь: "Продовжиться дія яка була"
- Коментар: В цілому вірно. Це називається "fall-through" — виконання перейде до наступного case без зупинки.
- Оцінка: 4/5.

8. Область видимості

- Ваша відповідь: "не можемо, бо змінна локальна"
- Коментар: Абсолютно вірно. Локальні змінні видимі лише у своєму блоці¹.
- Оцінка: 5/5.

9. Функції

- Ваша відповідь: "щоб не дублювати"
- Коментар: Правильно. Уникнення дублювання коду — головна мета.
- Оцінка: 5/5.

10. Геометрична логіка

- Ваша відповідь: "x має бути меншим" — НЕВІРНО.
 - Коментар: Давайте подивимось на слайд 24². Для "region2" (права, коричнева частина прямокутника) умова для x записана як $x > b/2$. Тобто x має бути більшим за середину, щоб точка потрапила у праву частину. "Меншим" — це для лівої частини (зеленої).
 - Оцінка: 0/5.
-

Підсумковий аналіз захисту

1. Ймовірність використання ШІ: Дуже низька.

Ваші відповіді дуже лаконічні, фрагментарні ("людський стиль") і містять фактичну помилку в 10-му питанні, якої ШІ, маючи перед "очима" текст лекції, швидше за все, не припустився б. Це свідчить про те, що ви відповідали самостійно.

2. Оцінювання:

- Частина 1 (Тести): 20 / 25 балів.
- Частина 2 (Відкриті питання): 16 / 25 балів (втрачено бали за неповну відповідь у №6 та помилку в №10).
- Сума: 36 / 50 балів.

Переведення у 5-балльну шкалу:

$$\$36 / 50 = 72\% \approx 3.6 \text{ (Оцінка: "Добре" / C)}$$

Вердикт викладача:

Ви захистили роботу. Ви добре розумієте базові принципи (функції, змінні), але є прогалини в уважності при роботі з умовами (геометрія) та трасуванні алгоритмів.

Наступний крок: Чи хочете ви, щоб я пояснив помилку в 10-му питанні детальніше (з малюнком координат), чи завершимо на цьому?

1. Уяви, що є вкладена конструкція if, але без фігурних дужок:

```
if (condition1)
```

```
if (condition2)
```

```
statement1;else
```

```
statement2;
```

До якого саме if належить else? До первого (condition1) чи до другого (condition2)?

2. Лініве обчислення

Ти задав мені питання про так зване лініве обчислення, можеш більше розказати чому так називається?

Звісно, я можу пояснити ці два важливі концепції, які часто викликають питання у тих, хто вивчає програмування.

1. Проблема "Висячого else" (Dangling else)

Проблема виникає через те, що фігурні дужки {} не використовуються для групування вкладених інструкцій if.

Відповідь:

У мові C++ (як і в більшості мов сімейства C) else завжди належить до найближчого попереднього if, для якого ще немає відповідного else.

У вашому прикладі:

C++

```
if (condition1)
    if (condition2) // Найближчий if без else
        statement1;
else // Це else належить до if (condition2)
    statement2;
```

Тобто statement2 буде виконано, лише якщо condition1 істинна, але condition2 хибна.

Як це розуміє компілятор (незалежно від форматування):

C++

```
if (condition1)
{
    if (condition2)
        statement1;
}
else // Відноситься до condition2
{
    statement2;
}
```

// Тут має бути код для ситуації, коли condition1 хибна, але його немає

Інженерне рішення:

Щоб уникнути неоднозначності та зробити код зрозумілим для людини (навіть якщо компілятор знає правило), завжди використовуйте фігурні дужки {} для групування тіл if та else .

2. "Лініве обчислення" (Short-circuit evaluation)

Термін "лініве обчислення" (або "коротке замикання") використовується, тому що компілятор "лінується" або уникає виконання зайвих дій, якщо результат логічного виразу вже визначений.

Чому так називається?

Причина криється в самій логіці операторів && (I) та || (АБО):

1. Для оператора && (AND):

- Щоб весь вираз був істинним (true), обидві його частини мають бути істинними.
- Якщо перша частина виявилася хибною (false), то весь вираз гарантовано буде хибним, незалежно від другої частини.
- Лінь компілятора: У цьому випадку компілятор не обчислює (не виконує) другу частину, щоб заощадити час.

2. Для оператора || (OR):

- Щоб весь вираз був істинним (true), достатньо, щоб хоча б одна частина була істинною.
- Якщо перша частина виявилася істинною (true), то весь вираз гарантовано буде істинним, незалежно від другої частини.
- Лінь компілятора: У цьому випадку компілятор не обчислює (не виконує) другу частину, оскільки результат вже відомий.

Ця "лінь" — це не просто оптимізація швидкості. Вона критично важлива для безпеки коду, особливо в прикладі з діленням на нуль, згаданому у

вашому попередньому питанні:

C++

```
if (denominator != 0 && (numerator / denominator) > 5)
```

Завдяки "лінівому обчисленню", якщо denominator дорівнює нулю, друга частина (numerator / denominator) ніколи не буде виконана, і програма уникне критичної помилки.

ВИСНОВОК

У процесі вивчення лекції та виконання роботи були засвоєні ключові принципи алгоритмізації та моделювання, включаючи властивості алгоритмів та їхнє представлення у вигляді діаграм активності UML. На практиці набуло навичок реалізації алгоритмів розгалуження та множинного вибору через інструкції if-else та switch у C++, з глибоким розумінням механізмів керування потоком, таких як break та "лініве обчислення". Також було вивчено важливість області видимості (scope) для локальних і глобальних змінних та значення функцій (void та з поверненням значення) для структурування та уникнення дублювання коду. Це дозволило успішно переводити складні логічні умови, зокрема для визначення положення точки у геометричній області, у коректну програмну реалізацію.