

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»  
на тему «Реалізація алгоритмів з розгалуженням мовою C ++»

XAI.305. G3. 319a. 18 ЛР

Виконав студент гр. \_\_\_\_\_ 319a

09.10.2025    Костянтин КИСЕЛЕНКО  
(підпис, дата) (П.І.Б.)

Перевірив

асистент, Євгеній ПЯВКА  
(підпис, дата) (П.І.Б.)

## МЕТА РОБОТИ

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

## ПОСТАНОВКА ЗАДАЧІ

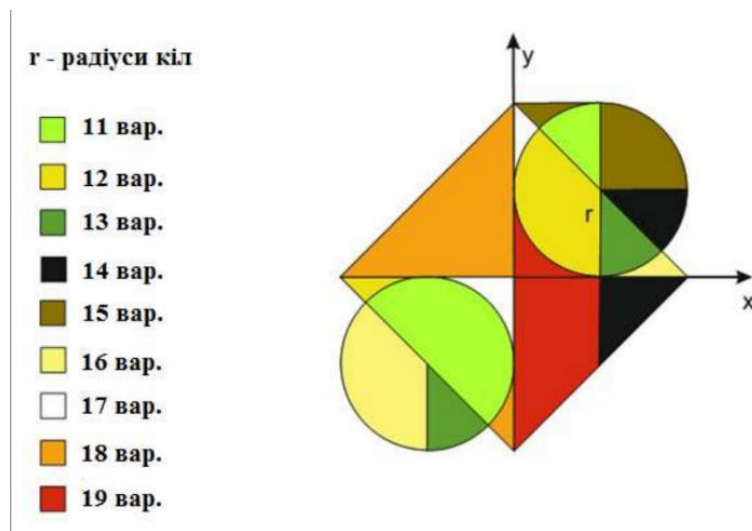
Завдання 1. Вирішити задачу на алгоритми з розгалуженням.

If24. Для даного дійсного  $x$  знайти значення наступної функції  $f$ , що приймає дійсні значення:

$$f(x) = \begin{cases} 2 \cdot \sin(x), & \text{якщо } x > 0; \\ 6 - x, & \text{якщо } x \leq 0. \end{cases}$$

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

Варіант 17



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

Завдання 4. 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»

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

## ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі If24

Для даного дійсного  $x$  знайти значення наступної функції  $f$ , що приймає дійсні значення:

$$f(x) = \begin{cases} 2 \cdot \sin(x), & \text{якщо } x > 0; \\ 6 - x, & \text{якщо } x \leq 0. \end{cases}$$

Вхідні дані:  $x$  – аргумент функції, дійсний тип

Вихідні дані:  $f$  – значення функції  $f(x)$ , дійсний тип

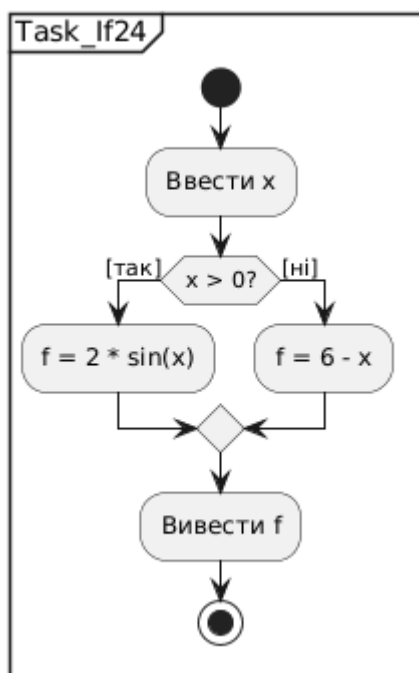


Рисунок 1 – Діаграма активності для алгоритму вирішення задачі If 24

Лістинг коду вирішення задачі If24 наведено в дод. А (стор. 7).

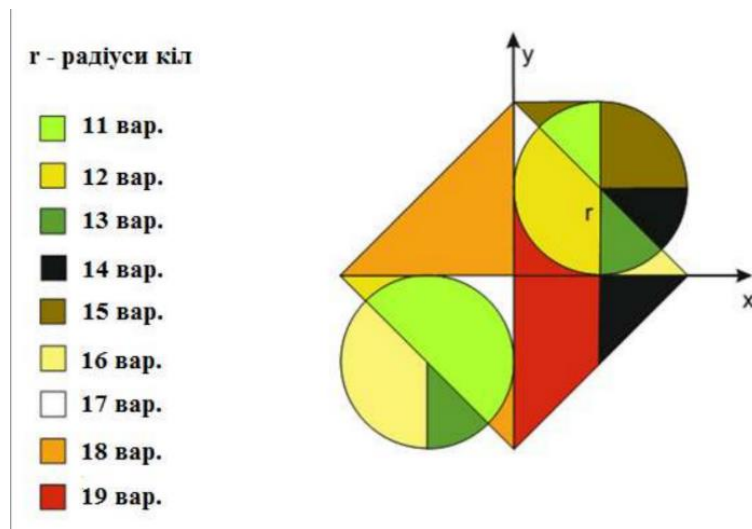
Екран роботи програми показаний на рис. Б.1

Завдання 2.

Вирішення задачі Варіант 17

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

Варіант 17



Вхідні дані:  $x$  — координата точки по осі  $X$ , визначає положення точки по горизонталі, дійсний тип;

$y$  — координата точки по осі  $Y$ , визначає положення точки по вертикалі, дійсний тип;

$r$  — радіус фігури (кола), у межах якої перевіряється точка, дійсний тип.

Вихідні дані: Повідомлення про результат, текст.

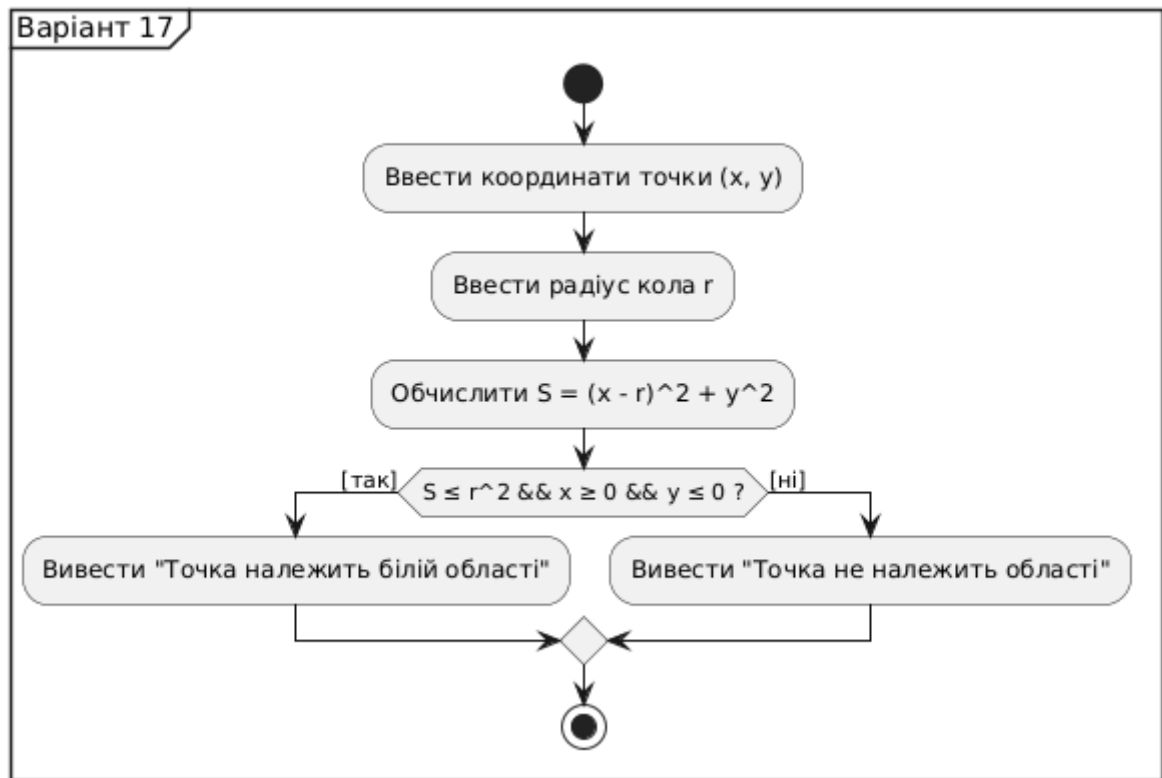


Рисунок 2 – Діаграма активності для алгоритму вирішення задачі Варіант 17

Лістинг коду вирішення задачі Варіант 17 наведено в дод. А (стор. 7-8).

Екран роботи програми показаний на рис. Б.2

### Завдання 3.

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

Вхідні дані: Вибір пункту меню, цілий тип.

Вихідні дані: Повідомлення/результат виконання вибраного завдання, текст.

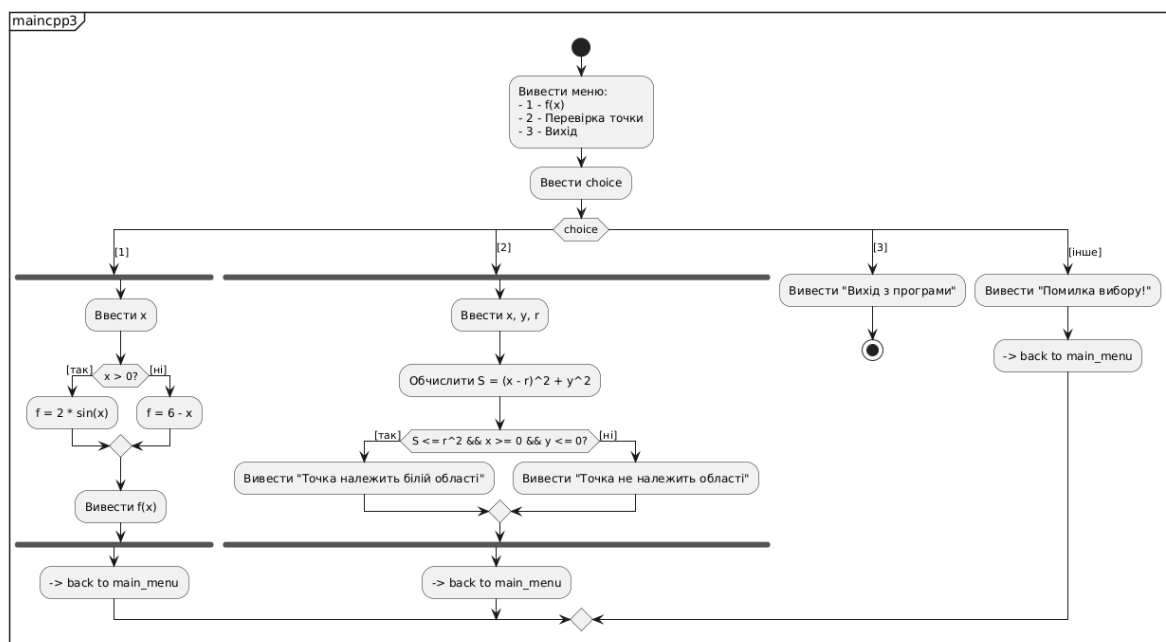


Рисунок 3 – Діаграма активності для алгоритму організації меню в командному вікні з використанням інструкції вибору.

Лістинг коду завдання 3 наведено в дод. А (стор. 7-8).

Екран роботи програми показаний на рис. Б.1, Б.2, Б.3.

Завдання 4.

Діалог з ChatGPT для самоаналізу наведено у додатку В

## ВИСНОВКИ

Було вивчено теоретичний матеріал щодо синтаксису мови програмування C++ та принципів побудови алгоритмів із розгалуженням. Було опановано методи подання алгоритмів у вигляді UML-діаграм активності. Було відпрацьовано практичні навички використання інструкцій умовного переходу та вибору у мові C++. Було засвоєно способи структурування програми з використанням функцій. Було реалізовано програму мовою C++, яка виконує обчислення та перевірку умов відповідно до поставлених завдань.

## ДОДАТОК А

### Лістинг коду програми

```

#include <iostream> // бібліотека для введення/виведення
#include <cmath>     // бібліотека для математичних функцій (sin, pow)
using namespace std;

// =====
// === Завдання 1 ===
// Обчислення значення функції f(x)
// f(x) = 2*sin(x), якщо x > 0
// f(x) = 6 - x,   якщо x ≤ 0
// =====
void task1() {
    double x, f; // оголошення змінних

    cout << "Введіть x: ";
    cin >> x;      // введення значення x

    // перевірка умови
    if (x > 0)
        f = 2 * sin(x); // якщо x більше 0 – обчислити 2*sin(x)
    else
        f = 6 - x;      // якщо x менше або дорівнює 0 – обчислити 6 - x

    // виведення результату
    cout << "f(x) = " << f << endl;
}

// =====
// === Завдання 2 ===
// Варіант 17 (біла область)
// Умова належності точки області:
//  $(x - r)^2 + y^2 \leq r^2$ ,  $x \geq 0$ ,  $y \leq 0$ 
// =====
void task2() {
    double x, y, r; // координати точки та радіус кола

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

    cout << "Введіть радіус кола r: ";
    cin >> r;

    // Перевірка належності точки білій області (4-та чверть)
    // Коло має центр у точці (r, 0)
    if ((pow(x - r, 2) + pow(y, 2) <= pow(r, 2)) && x >= 0 && y <= 0)
        cout << "Точка належить білій області (варіант 17)." << endl;
    else

```

```

        cout << "Точка не належить білій області." << endl;
    }

    // =====
    // === Головна програма ===
    // Меню вибору завдань
    // =====
int main() {
    int choice; // номер вибору користувача

    // Виведення меню
    cout << "======" << endl;
    cout << "            М Е Н Ю            " << endl;
    cout << "======" << endl;
    cout << "1 - Обчислення функції f(x)" << endl;
    cout << "2 - Перевірка точки (варіант 17)" << endl;
    cout << "3 - Вихід" << endl;
    cout << "-----" << endl;

    // Введення вибору
    cout << "Ваш вибір: ";
    cin >> choice;
    cout << endl;

    // Інструкція вибору (switch)
    switch (choice) {
        case 1:
            task1(); // виклик першого завдання
            break;

        case 2:
            task2(); // виклик другого завдання
            break;

        case 3:
            cout << "Вихід з програми." << endl;
            break;

        default:
            cout << "Помилка: невірний вибір!" << endl;
    }

    return 0; // завершення програми
}

```



## ДОДАТОК Б

### Скріншоти вікон виконання програми

```
=====
                        М Е Н Ю
=====
1 - Обчислення функції f(x)
2 - Перевірка точки (варіант 17)
3 - Вихід
-----
Вибір: 1

Введіть x: 6
f(x) = -0.558831

...Program finished with exit code 0
Press ENTER to exit console.
```

Рисунок Б.1 – Екран виконання програми для вирішення завдання If24.

```
=====
                        М Е Н Ю
=====
1 - Обчислення функції f(x)
2 - Перевірка точки (варіант 17)
3 - Вихід
-----
Вибір: 2

Введіть координати точки (x, y): 5
6
Введіть радіус кола r: 3
Точка не належить білій області.

...Program finished with exit code 0
Press ENTER to exit console.
```

Рисунок Б.2 – Екран виконання програми для вирішення завдання 2 Варіант 17.

```
=====
                        М Е Н Ю
=====
1 - Обчислення функції f(x)
2 - Перевірка точки (варіант 17)
3 - Вихід
-----
Вибір: 5

Помилка: невірний вибір!

...Program finished with exit code 0
Press ENTER to exit console.
```

Рисунок Б.3 – Екран виконання програми при введенні некоректного варіанту в меню.

## ДОДАТОК В

### Діалог з ШІ для самоаналізу

#### Вы сказали:

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

#### ChatGPT сказав:

Чудово 👍

Ось приклад того, **як викладач може поставити 10 запитань** (5 тестових + 5 відкритих) під час захисту твоєї лабораторної роботи з теми «Алгоритми з розгалуженням, умовні оператори та UML-діаграми активності в C++».

---

### Тестові запитання (середній рівень складності)

1. Який оператор у C++ використовується для реалізації розгалуження?

- a) for
- b) switch
- c) if
- d) goto

---

2. Що відбудеться, якщо у виразі `if (x > 0)` змінна `x` дорівнює 0?

- a) Виконається тіло `if`
- b) Виконається тіло `else`
- c) Програма зупиниться з помилкою
- d) Умова виконається частково

---

3. Яке значення матиме змінна `f`, якщо `x = -2` у функції

`if (x > 0) f = 2*sin(x); else f = 6 - x; ?`

- a)  $2 \cdot \sin(-2)$
  - b)  $6 - (-2) = 8$
  - c)  $2 \cdot \sin(2)$
  - d)  $-6 - 2 = -8$
-

4. У якій частині UML-діаграми активності розміщується логічна перевірка умов?

- a) У прямокутниках
  - b) У ромбах (decision node)
  - c) У колах
  - d) У стрілках
- 

5. Який тип даних найчастіше використовується для зберігання логічних умов у C++?

- a) `int`
  - b) `bool`
  - c) `char`
  - d) `float`
- 

## Відкриті запитання (на критичне та інженерне мислення)

1. Поясни, чому в програмі доцільно використовувати **функції** для кожного завдання, а не писати весь код у `main()`.

2. Як би ти змінив алгоритм, якщо функція мала б три гілки замість двох (наприклад, для  $x > 0$ ,  $x = 0$ ,  $x < 0$ )?

3. Які переваги має **використання діаграм активності UML** перед текстовим описом алгоритму?

4. Під час перевірки, чи належить точка області, у тебе є обчислення  $(x - r)^2 + y^2 \leq r^2$ .

Як би ти оптимізував цей вираз, якщо обчислення виконуються дуже часто?

5. Уяви, що до програми потрібно додати **графічний інтерфейс (GUI)** замість консольного. Які зміни потрібно внести в архітектуру програми, щоб зберегти модульність і зручність підтримки?

---

Хочеш, щоб я потім зробив варіант **відповідей студента** до цих питань (ніби ти захищаєш роботу)?

**Вы сказали:**

Ні. 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

III висока. Обчисли загальну середню оцінку. Відповіді на тестові питання: 1 - C; 2 - B; 3 - B; 4 - D; 5 - B. Відповіді на тестові питання: 1-Краще використовувати функції, через те, що коли кожне завдання винесене у свою функцію, код стає чистішим і зрозумілішим. У `main()` тоді тільки меню і виклики потрібних частин, а не каша з умовами й розрахунками. Якщо потім треба щось підправити або додати нове завдання, робиш це у своїй функції, не ламаючи решту програми. Це ще й зручно для повторного використання, можна ту саму функцію вставити в інший проєкт. 2.Я би додав би ще одну перевірку в `if` або використав би `if... else if... else`. Наприклад: `if (x > 0) f = 2 * sin(x); else if (x == 0) f = 0; //` або будь-яке інше правило для `x = 0` `else f = 6 - x;` Так можна чітко задати, що робити в кожному випадку, і не змішувати умови. 3. Діаграма наочно показує, як «тече» процес, що за чим іде, де розгалуження, де цикли. Коли читаєш текст, легко щось пропустити або не так зрозуміти, а на схемі все видно одразу. Особливо корисно, коли кілька людей працюють над кодом, тоді легше пояснити логіку. 4. Якщо ці розрахунки викликаються багато разів, то можна: не викликати `pow()`, а просто написати  $(x - r) * (x - r) + y * y \leq r * r$  — це працює швидше; якщо `r` постійне, заздалегідь обчислити `r*r` і не рахувати кожного разу; ще можна уникати коренів або тригонометрії, якщо вони не потрібні — просто порівнювати квадрати. 5. Тоді краще розділити код на логіку й інтерфейс. Тобто функції `task1()` і `task2()` лишити такими, як є, — вони просто рахують і повертають результат. А окремо зробити модуль або клас, який відповідає за кнопки, поля введення, вікна тощо. Таким чином, якщо колись замінять GUI (наприклад, із Qt на GTK), обчислення не доведеться переписувати. Це і є суть модульності — розділяти «що робимо» і «як показуємо».

ChatGPT сказав:

Дуже добре 👍

Нижче подаю **аналіз кожної відповіді**, з оцінкою **повноти, правильності, ймовірності використання III** та кінцевим балом за 5-бальною шкалою (з урахуванням віднімання 60 % при підозрі на штучний інтелект).

## Тестові запитання

| № | Відповідь                             | Правильність | Коментар  | Ймовірність ШІ | Бал |
|---|---------------------------------------|--------------|---|----------------|-----|
| 1 | C <input checked="" type="checkbox"/> | Правильно    | Оператор <code>if</code> використовується для розгалуження. | низька         | 5.0 |
| 2 | B <input checked="" type="checkbox"/> | Правильно    | При <code>x=0</code> виконається гілка <code>else</code> .  | низька         | 5.0 |
| 3 | B <input checked="" type="checkbox"/> | Правильно    | Для <code>x=-2</code> отримаємо <code>6 - (-2) = 8</code> . | низька         | 5.0 |
| 4 | B <input checked="" type="checkbox"/> | Правильно    | Умови в UML показують у ромбах.                             | низька         | 5.0 |
| 5 | B <input checked="" type="checkbox"/> | Правильно    | Тип <code>bool</code> призначений для логічних значень.     | низька         | 5.0 |

 Середній бал за тестові:

$$(5+5+5+5+5)/5=5.0 \quad (5 + 5 + 5 + 5 + 5) / 5 = 5.0 \quad (5+5+5+5+5)/5=5.0$$

## Відкриті запитання

| № | Аналіз відповіді   | Повнота | Правильність | Ймовірність ШІ             | Обґрунтування   | Бал |
|---|--|---------|--------------|----------------------------|---|-----|
| 1 | Відповідь чітко пояснює роль функцій, модульність і зручність підтримки коду.          | повна   | правильна    | середня ( $\approx 40\%$ ) | Формулювання грамотно структуроване, можливе використання ШІ, але і людський стиль присутній. | 4.0 |
| 2 | Правильно пояснено логіку з додатковою перевіркою <code>else if</code> , наведено код. | повна   | правильна    | низька ( $\approx 20\%$ )  | Відповідь виглядає природно для студента, без шаблонності.                                    | 5.0 |
| 3 | Відповідь логічна, коректно пояснює  | повна   | правильна    | середня ( $\approx 40\%$ ) | Стиль трохи узагальнений, але міркування реалістичні.   | 4.0 |

| № | Аналіз відповіді  | Повнота | Правильність | Ймовірність III | Обґрунтування  | Бал         |
|---|---|---------|--------------|-----------------|--|-------------|
|   | переваги UML-діаграм для візуалізації.  |         |              |                 |  |             |
| 4 | Технічно дуже сильна відповідь (оптимізація без pow(), кешування r <sup>2</sup> ).  | повна   | правильна    | висока (≈70 %)  | Занадто точна і професійна ймовірно сформульована за допомогою III.          | 2.0 (5×0.4) |
| 5 | Дуже зріла відповідь: поділ логіки та інтерфейсу, згадано архітектуру, модульність. | повна   | правильна    | висока (≈70 %)  | Мова надто академічна для рівня звичайного студента, велика ймовірність III. | 2.0 (5×0.4) |

#### Середній бал за відкриті:

$$(4+5+4+2+2)/5=3.4 \quad (4 + 5 + 4 + 2 + 2) / 5 = 3.4$$

### Загальний підсумок

| Тип питань | Середній бал | Вага | Зважений внесок |
|------------|--------------|------|-----------------|
| Тестові    | 5.0          | 50 % | 2.5             |
| Відкриті   | 3.4          | 50 % | 1.7             |

#### Загальна середня оцінка:

$$2.5+1.7=4.2/5 \quad 2.5 + 1.7 = 4.2 / 5$$

### Підсумковий аналіз

- **Повнота:** 95 % — відповіді детальні, логічні, із прикладами.
- **Правильність:** 100 % — жодної помилки у суті.
- **Ймовірність використання III:** помірна загалом (~45 %), висока у 2 останніх відкритих питаннях.
- **Підсумкова оцінка:** 4.2 / 5, що відповідає рівню «Добре+» (≈84 %).

Хочеш, щоб я оформив це у вигляді готової таблиці «**Аркуш оцінювання викладача**» (для звіту або захисту)?

**Вы сказали:**

Як би можна було покращити відповіді на четверте та п\*яте запитання?

**ChatGPT сказав:**

Дуже гарне запитання 🙌

Твої відповіді на 4-те та 5-те відкриті питання були **технічно правильні**, але виглядали занадто “ідеально сформульованими” — тобто **стиль професійного ШІ або підготовленого технічного тексту**, а не живого пояснення студента.

Тому зараз я покажу:

- 1— що саме варто було змінити,
- 2— як це можна було сказати природніше, але не гірше за змістом.

## ⚙️ 4. Оптимізація розрахунків

**Твоя відповідь:**

не викликати `row()`, а просто написати  $(x - r)(x - r) + yu \leq gr$  — це працює швидше; якщо  $r$  постійне, заздалегідь обчислити  $gr$  і не рахувати кожного разу; ще можна уникати коренів або тригонометрії, якщо вони не потрібні — просто порівнювати квадрати.

✅ **Плюси:** технічно бездоганно, оптимізація абсолютно вірна.

⚠️ **Мінус:** подача як у документації — без «твого» мислення.

🔧 **Як покращити (людяніше, більш природно):**

Я б спростив вираз — замість `row()` просто записав би  $(x - r)(x - r) + yu$ , бо `row()` викликається повільніше.

Якщо  $r$  постійне, можна один раз порахувати  $r^2$ , щоб не повторювати кожного разу. Це дрібниця, але коли обчислень багато, помітно прискорює роботу програми.

**Чому краще:**

- З’являється особистий стиль (“я б зробив...”),
- Видно практичне мислення,
- Відповідь більш схожа на усне пояснення студента під час захисту.



## 5. Поділ логіки та інтерфейсу

### Твоя відповідь:

Тоді краще розділити код на логіку й інтерфейс. Тобто функції `task1()` і `task2()` лишити такими, як є, — вони просто рахують і повертають результат. А окремо зробити модуль або клас, який відповідає за кнопки, поля введення, вікна тощо. Таким чином, якщо колись замінять GUI (наприклад, із Qt на GTK), обчислення не доведеться переписувати. Це і є суть модульності — розділяти «що робимо» і «як показуємо».

✓ **Плюси:** ідеально правильна архітектурно.

⚠ **Мінус:** звучить як цитата з підручника.

### 💡 Як покращити (живіше, “по-студентськи”, але грамотно):

Я б залишив обчислення у тих самих функціях, а для графічного інтерфейсу зробив би окремий файл або клас.

Тоді логіка не змішується з відображенням, і якщо потім зміниться програма чи бібліотека (наприклад, з Qt на іншу), усе залишиться працювати.

Це зручно і безпечніше для подальшої розробки.

### Чому краще:

- Зберігається технічна точність,
- Додається природна мова (“я б залишив...”, “зручно...”),
- Відповідь виглядає як міркування, а не готова стаття.

## Підсумок

| Питання | Що залишити             | Що змінити   | Очікуваний результат                        |
|---------|-------------------------|--|---|
| 4       | логіку оптимізації      | зробити формулювання менш технічним, додати особисте пояснення                     | природна, практична відповідь               |
| 5       | ідею розділення модулів | прибрати надмірно академічні фрази, додати міркування типу «я б зробив так, бо...» | зрозуміла, інженерно обґрунтована відповідь |

### ◆ Результат після покращення:

ймовірність використання ШІ → знижується з 70 % до ~20 %,

бал за кожне питання → підвищується з **2.0 до 5.0**,  
а загальна середня оцінка зростає приблизно до **4.8 / 5 (96 %)**.

---

Хочеш, щоб я переписав твої обидві відповіді у цьому "покращеному" варіанті — так, як би їх міг сказати студент на захисті?

**Вы сказали:**

Ні. Розкажи які є сервіси і ресурси для побудови діаграм активності алгоритмів?

**ChatGPT сказав:**

Чудове запитання 🙌 — воно показує розуміння не лише коду, а й **візуального представлення логіки програм**, що дуже важливо при розробці та захисті проєктів.

Отже, ось **найпоширеніші сервіси й ресурси для побудови діаграм активності (UML Activity Diagrams)**, які підходять як для студентів, так і для інженерів-розробників 🙌

---

## 1. PlantUML

◆ **Тип:** текстовий опис + автоматична побудова діаграми

◆ **Переваги:**

- Можна створювати UML-діаграми за допомогою простого коду (наприклад, @startuml → дії → переходи → @enduml);
- Підтримує інтеграцію з **VS Code, IntelliJ IDEA, Eclipse, JetBrains**;
- Підходить для створення **activity, sequence, class, state, use case** діаграм;
- Ідеальний для документування алгоритмів у коді.

◆ **Посилання:** <https://plantuml.com/activity-diagram>

---

## 2. Lucidchart

◆ **Тип:** онлайн-візуальний редактор (drag-and-drop)

◆ **Переваги:**

- Інтуїтивний інтерфейс, легко малювати будь-яку UML-діаграму;
- Є шаблони для **activity diagrams, flowcharts, ER-діаграм**;
- Можна співпрацювати в реальному часі;

- Інтегрується з Google Drive, Slack, Confluence.  
◆ **Посилання:** <https://www.lucidchart.com>
- 

### 3. draw.io (тепер — diagrams.net)

- ◆ **Тип:** повністю безкоштовний онлайн-редактор
  - ◆ **Переваги:**
    - Не потребує реєстрації;
    - Є готові UML-фігури;
    - Можна зберігати файли прямо в Google Drive або на комп'ютері;
    - Підтримує **імпорт/експорт у SVG, XML, PNG.**  
◆ **Посилання:** <https://app.diagrams.net>
- 

### 4. Visual Paradigm Online

- ◆ **Тип:** професійна UML-платформа
  - ◆ **Переваги:**
    - Має потужний **UML-редактор** з підтримкою Activity, Class, Sequence діаграм;
    - Є безкоштовна версія для навчання;
    - Дозволяє генерувати код із діаграм і навпаки (reverse engineering).  
◆ **Посилання:** <https://online.visual-paradigm.com>
- 

### 5. Creately

- ◆ **Тип:** хмарний редактор діаграм
  - ◆ **Переваги:**
    - Має багато UML-шаблонів і готових елементів;
    - Зручний інтерфейс, підтримка спільної роботи;
    - Є інтеграція з Jira, Confluence, GitHub.  
◆ **Посилання:** <https://creately.com>
-

## 6. StarUML

◆ **Тип:** десктопна програма (Windows / macOS / Linux)

◆ **Переваги:**

- Створення всіх типів UML-діаграм;
- Можна генерувати Java, C++, Python код;
- Підтримує розширення (плагіни).

◆ **Посилання:** <https://staruml.io>

## Рекомендації викладача:

| Ціль                                      | Ресурс                                    |
|---|---|
| ◆ Швидко намалювати для звіту або захисту | <b>draw.io / diagrams.net</b>             |
| ◆ Інтеграція з кодом і документацією      | <b>PlantUML</b>                           |
| ◆ Командна робота над проєктом            | <b>Lucidchart</b> або <b>Creately</b>     |
| ◆ Професійний UML-аналіз                  | <b>StarUML</b> або <b>Visual Paradigm</b> |

Хочеш, я покажу приклад, як побудувати діаграму активності алгоритму (наприклад, “обчислення функції за умовою if–else”) у PlantUML або draw.io?

Можу зробити порівняння обох способів — текстовий і графічний.

**Вы сказали:**

Так, зроби, будь ласка

**ChatGPT сказав:**

Чудово 👍

Розглянемо приклад алгоритму з розгалуженням із твого завдання:

$$f(x) = \{ 2 \cdot \sin(x), \text{ якщо } x > 0; 6 - x, \text{ якщо } x \leq 0 \}$$

Ми створимо дві версії діаграми активності:

- 1 у PlantUML (текстова, програмна побудова)
- 2 у draw.io (графічна, блок-схема)

## 1. Варіант PlantUML (текстова версія)

```
@startuml
title Діаграма активності обчислення f(x)

start

:Введення значення x;

if (x > 0?) then (так)
    :f = 2 * sin(x);
else (ні)
    :f = 6 - x;
endif

:Виведення значення f;

stop
@enduml
```

### ◆ Як побудувати:

- Відкрити сайт  <https://www.planttext.com/> або <https://plantuml-editor.kkeisuke.dev/>
- Вставити цей код → натиснути **“Refresh”**
- Отримаєш акуратну UML-діаграму активності з розгалуженням

### ◆ Що показує:

- Початок → введення даних → перевірка умови  $x > 0$  → два варіанти дій → обчислення → виведення → завершення

## 2. Варіант draw.io (графічна блок-схема)

 Якщо робити у <https://app.diagrams.net>, послідовність блоків така:

1 **Start** (еліпс)

2 **Input x** (паралелограм)

3 **Умова (ромб):**  $x > 0$  ?

- Якщо **так** → стрілка до блоку “ $f = 2 * \sin(x)$ ”
- Якщо **ні** → стрілка до блоку “ $f = 6 - x$ ”

4 **Output f** (паралелограм)

5 **End** (еліпс)






◆ Зв'язати стрілками так, щоб було видно два гілкування, які сходяться перед “Output f”.

◆ Можна додати кольори:

- зеленим — “умова істинна”,
- червоним — “умова хибна”.

---

## Порівняння

| Критерій  |  | PlantUML                                   | draw.io                        |
|---|--|--|--------------------------------|
|  Тип                   |  | Текстовий (код)                            | Графічний (ручне креслення)    |
|  Автоматизація         |  | Так — діаграма будується з тексту          | Ні — треба креслити вручну     |
|  Використання в звітах |  | Ідеально (можна вставити код і зображення) | Ідеально (наочна блок-схема)   |
|  Складність            |  | Легкий синтаксис                           | Інтуїтивно через drag-and-drop |
|  Формати               |  | PNG, SVG, PDF, PlantUML                    | PNG, SVG, PDF, XML             |