

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.301.G12.311.2 ЛР

Виконав студент гр. 311

12.11.2025

(підпис, дата)

Ростислава БОЖОНОК

(П.І.Б.)

Перевірів

к.т.н., доц. Олена ГАВРИЛЕНКО

(підпис, дата)

(П.І.Б.)

МЕТА РОБОТИ

Ознайомитися з умовними операторами мови C++, реалізувати алгоритми з розгалуженням у вигляді окремих функцій, побудувати UML-діаграми активності, перевірити коректність введених даних, та провести самоаналіз знань за допомогою ІІІ.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити дві задачі на алгоритми з розгалуженням.

If38-табл.1 Для заданого дійсного x знайти значення наступної функції r , що приймає значення цілого типу:
$$r(x) = \begin{cases} x^2 - 4, & \text{якщо } x < -2; \\ x, & \text{якщо } -2 \leq x \leq 2; \\ 4, & \text{якщо } x > 2. \end{cases}$$

If14-табл.1 Дано три числа. Вивести спочатку найменше, а потім найбільше з даних чисел.

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

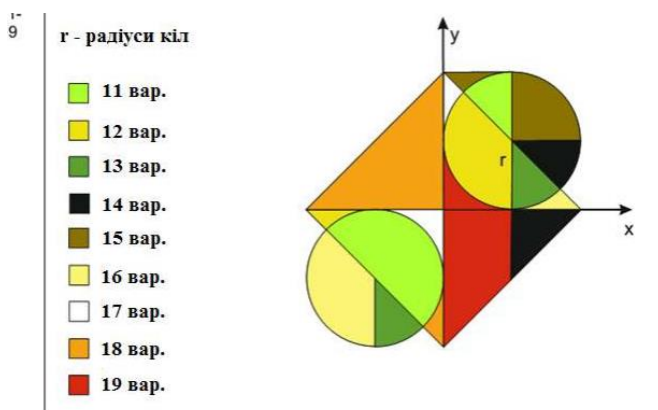


Рисунок 1 – Геометрична задача 18 варіант

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі If38

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

x - дійсне число (тип double), довільне значення.

Вихідні дані (ім'я, опис, тип):

r – результат обчислення функції $r(x)$, тип double

Алгоритм вирішення показано нижче:

1. Вивести запит на введення значення x .
2. Зчитати значення x .
3. Перевірити, чи введено коректне число.
4. Якщо $x < -2$, обчислити $r = x^2 - 4$.
5. Інакше, якщо $x \leq 2$, обчислити $r = x$.
6. Інакше (тобто $x > 2$), присвоїти $r = 4$.
7. Вивести результат r .

Лістинг коду вирішення задачі розділ і номер задачі наведено в дод. А (стор.6).

Завдання 2.

Вирішення задачі If14

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

a, b, c – три дійсних числа (тип double), довільне значення.

Вихідні дані (ім'я, опис, тип):

$\min v$ – найменше з трьох чисел (тип double)

$\max v$ – найбільше з трьох чисел (тип double)

Алгоритм вирішення показано нижче:

1. Вивести запит на введення трьох чисел a, b, c .
2. Зчитати значення a, b, c .
3. Перевірити коректність вводу.
4. Ініціалізувати $\min v = a, \max v = a$.
5. Якщо $b < \min v$, то $\min v = b$.
6. Якщо $c < \min v$, то $\min v = c$.
7. Якщо $b > \max v$, то $\max v = b$.

8. Якщо $c > \max v$, то $\max v = c$.
9. Вивести $\min v$ та $\max v$.

Лістинг коду вирішення задачі розділ і номер задачі наведено в дод. А (стор. 6).

Завдання 3.

Вирішення задачі 18(геомет.)

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

a, b – додатні дійсні числа (тип double), параметри фігури, $0 < a < \frac{b}{2}$

x, y – координати точки (тип double)

Вихідні дані (ім'я, опис, тип):

Текстове повідомлення про належність точки області (тип string)

Алгоритм вирішення показано нижче:

1. Ввести параметри a, b .
2. Перевірити коректність: якщо $a \leq 0$ або $b \leq 0$ або $a \geq b/2$, то вивести "Wrong data!" і завершити.
3. Ввести координати точки (x, y) .
4. Перевірити коректність вводу.
5. Обчислити:
6. Якщо точка в прямокутнику справа ($x > b/2, x < b, y > 0, y < a$) і поза кругом ($\text{circle} > R^2$), або точка в півколі зліва ($\text{circle} < R^2, x < b/2$), то вивести "Точка належить області".
7. Інакше — "Точка НЕ належить області".

Лістинг коду вирішення задачі розділ і номер задачі наведено в дод. А (стор. 6).

Екрани робіт програм показаний на рис. Б. 9-14

ВИСНОВКИ

У ході лабораторної роботи було закріплено знання про умовні оператори мови C++. Реалізовано три задачі з розгалуженням у вигляді окремих функцій, організовано меню вибору задач. Відпрацьовано навички перевірки коректності введених даних та побудови діаграм активності. Отримано практичний досвід структурованого програмування.

ДОДАТОК А

Лістинг коду програми

```

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

// декларації функцій
void task_if38();
void task_if14();
void task_geom18();

int main() {
    int menu;
    cout << "Task number: ";
    cin >> menu;

    // перемикання між задачами
    switch (menu) {
        case 1: task_if38(); break;
        case 2: task_if14(); break;
        case 3: task_geom18(); break;
        default: cout << "Wrong task! (Only 1,2,3)" << endl;
    }

    system("pause");
    return 0;
}

// If38. Для заданого дійсного x знайти r(x):
// r(x) = x^2 - 4, якщо x < -2;
// r(x) = x, якщо -2 ≤ x ≤ 2;
// r(x) = 4, якщо x > 2.
void task_if38() {
    double x;
    cout << "***** If38 *****" << endl;
    cout << "x = ";
    cin >> x;

    if (!cin) {
        cout << "Wrong input!" << endl;
        return;
    }

    double r;
    if (x < -2)
        r = x * x - 4;

```

```

    else if (x <= 2)
        r = x;
    else
        r = 4;

    cout << "r(x) = " << r << endl;
    return;
}

// If14. Дано три числа. Вивести найменше, потім найбільше.
void task_if14() {
    double a, b, c;
    cout << "***** If14 *****" << endl;
    cout << "Enter a, b, c: ";
    cin >> a >> b >> c;

    if (!cin) {
        cout << "Wrong input!" << endl;
        return;
    }

    double minv = a, maxv = a;
    if (b < minv) minv = b;
    if (c < minv) minv = c;
    if (b > maxv) maxv = b;
    if (c > maxv) maxv = c;

    cout << "Min = " << minv << endl;
    cout << "Max = " << maxv << endl;
    return;
}

// Геометр.18. Перевірити, чи точка належить області (приклад з методички)
void task_geom18() {
    double a, b, x, y;
    cout << "***** Геометр.18 *****" << endl;
    cout << "Enter a, b: ";
    cin >> a >> b;

    if (cin.fail() || b/2 < a || a <= 0 || b <= 0) {
        cout << "Wrong data!" << endl;
        return;
    }

    cout << "Enter x, y: ";
    cin >> x >> y;
    if (!cin) {
        cout << "Wrong input!" << endl;
        return;
    }
}

```

```
double R2 = a * a / 4.0;
double xc = b / 2.0, yc = a / 2.0;
double circle = pow(x - xc, 2) + pow(y - yc, 2);

bool inRectExCirc = x > b/2 && x < b && y > 0 && y < a && circle > R2;
bool inHalfCirc = circle < R2 && x < b/2;

if (inRectExCirc || inHalfCirc)
    cout << "Точка належить області 1 (темна частина)." << endl;
else
    cout << "Точка НЕ належить області 1." << endl;

return;
}
```


ДОДАТОК Б

Скрін-шоти вікна виконання програми

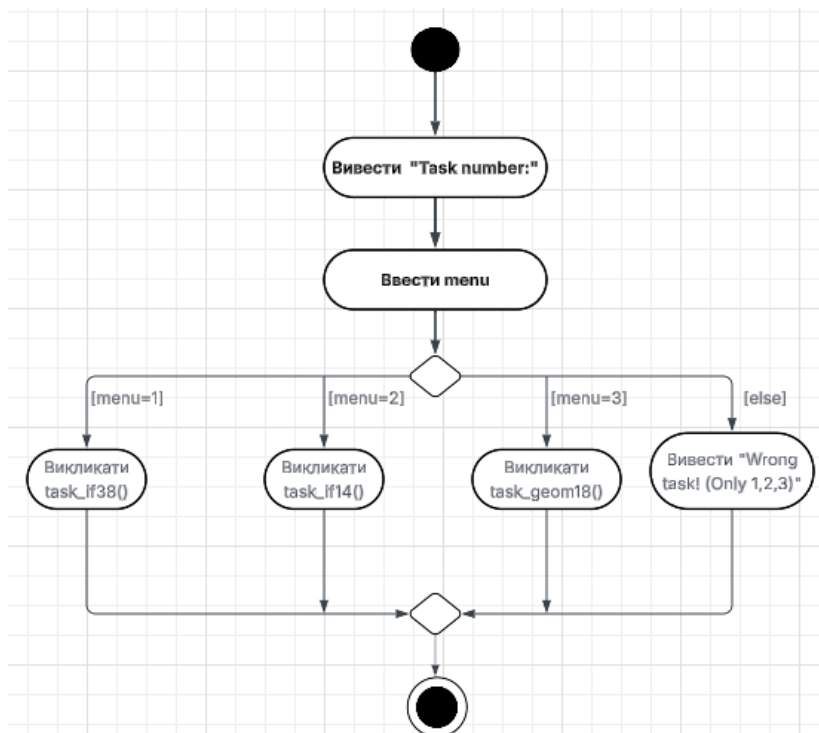


Рисунок Б.1 - Діаграма активності для всієї програми

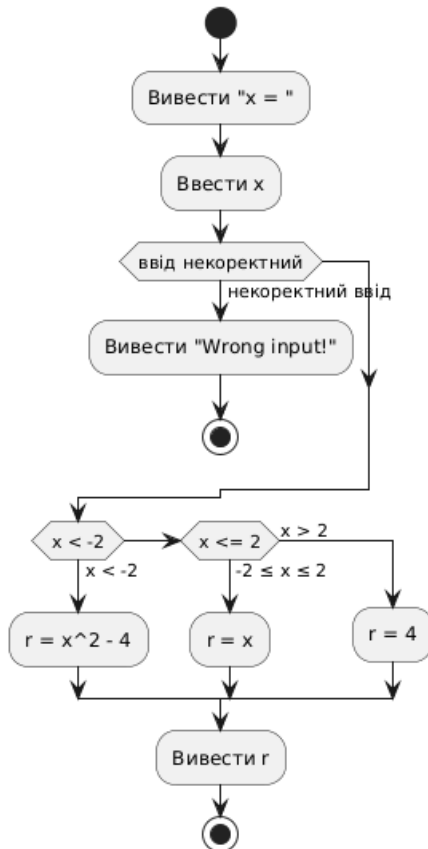


Рисунок Б.2 - Діаграма активності для алгоритму вирішення задачі If38



Рисунок Б.3 – Екран виконання програми для вирішення задачі If38

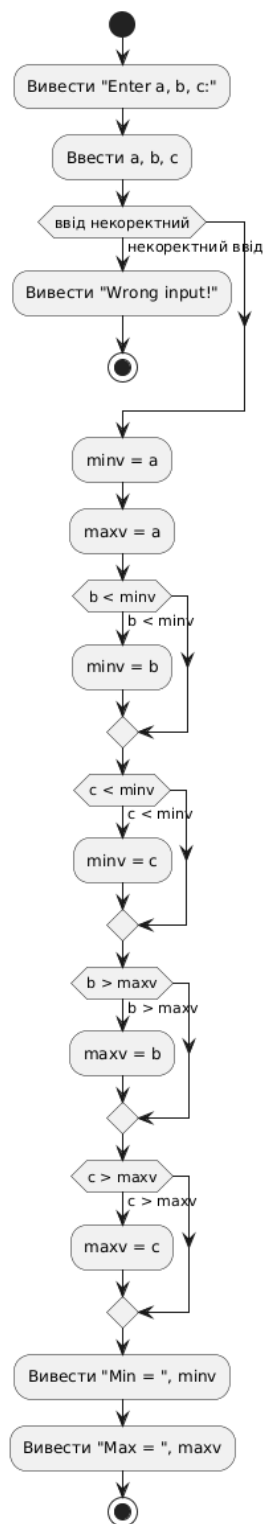


Рисунок Б.4 – Діаграма активності для алгоритму вирішення задачі If14

```

19      case 3: task_geom18(); break;
Task number: 2
***** If14 *****
Enter a, b, c: f b p
Wrong input!
sh: 1: pause: not found

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

19      case 3: task_geom18(); break;
Task number: 2
***** If14 *****
Enter a, b, c: 3 3 3
Min = 3
Max = 3
sh: 1: pause: not found

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

69      if (b > maxv) maxv = b;
70      if (c > maxv) maxv = c;
71
Task number: 2
***** If14 *****
Enter a, b, c: 56 34 19
Min = 19
Max = 56
sh: 1: pause: not found

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

```

Рисунок Б.5 – Екран виконання програми для вирішення задачі If14

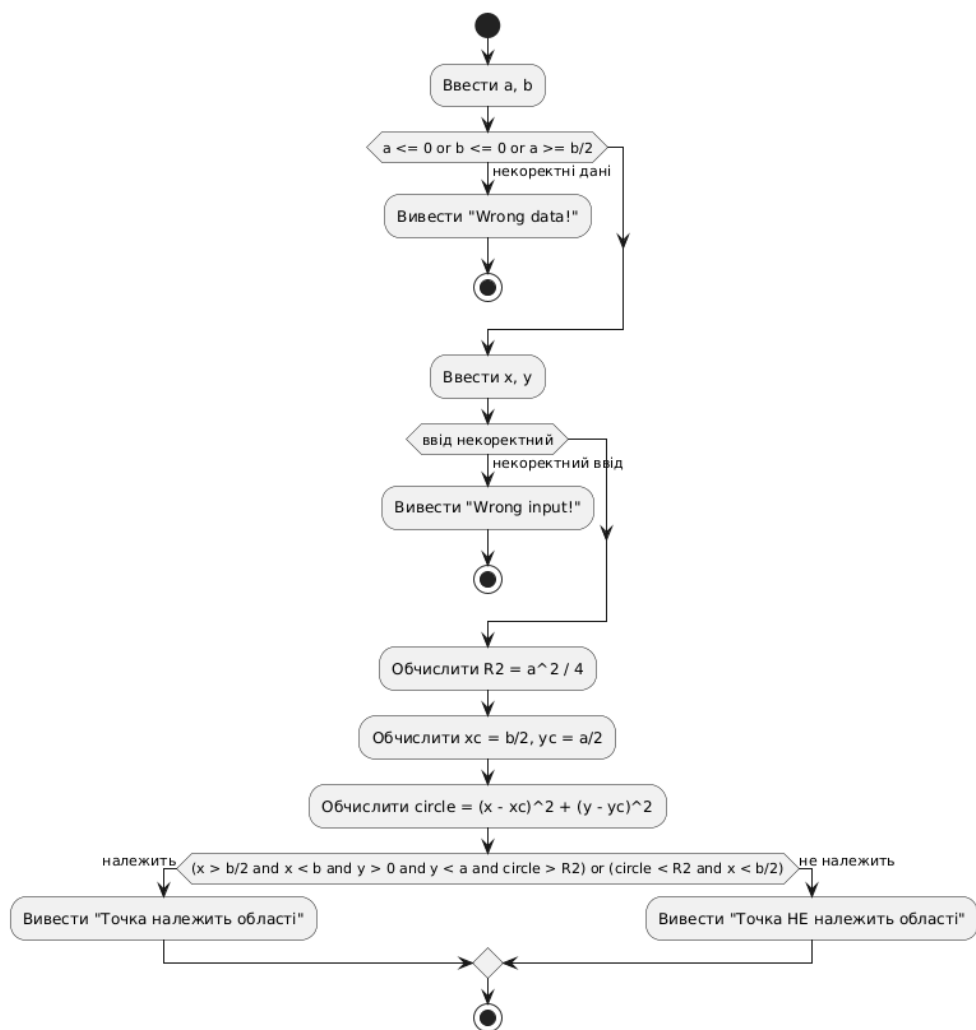


Рисунок Б.6 – Діаграма активності для алгоритму вирішення задачі task_geom18

The image consists of three screenshots of a terminal window showing the execution of a C++ program for task_geom18. Each screenshot has a code editor at the top and a terminal output below.

First Screenshot: The code editor shows line 19: `case 3: task_geom18(); break;`. The terminal output shows:


```
Task number: 3
***** Геометр.18 *****
Enter a, b: c a
Wrong data!
sh: 1: pause: not found

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

Second Screenshot: The code editor shows the same line 19. The terminal output shows:


```
Task number: 3
***** Геометр.18 *****
Enter a, b: 4 6
Wrong data!
sh: 1: pause: not found

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

Third Screenshot: The code editor shows lines 92 and 93: `cout << "Wrong input!" << endl;` and `return;`. The terminal output shows:


```
Task number: 3
***** Геометр.18 *****
Enter a, b: 9 42
Enter x, y: 2 4
Точка НЕ належить області 1.
sh: 1: pause: not found

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

Рисунок Б.7 – Екран виконання програми для вирішення задачі task_geom18

ДОДАТОК В

Діалог з ІІІ для самоаналізу

CPP_Тема 6_мас.pdf

PDF

CPP_Лекц 6 мас.pdf

PDF

Лабораторна робота № 3.docx

Документ

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:

Гаразд, як викладач на захисті, я підготував **5 тестових і 5 відкритих** запитань **середнього рівня**, які стосуються:

- твоєї програми з ЛР №3 (функції `task_if38()`, `task_if14()`, `task_geom18()`)
- теорії з лекції та методички (масиви, умови, передача параметрів, алгоритми)

✓ Тестові питання (5 шт.)

1. У функції `task_if38()` вираз `else if (x <= 2)` означає:

- A. Перевірку лише $x = 2$
- B. Перевірку всіх значень x менше -2
- C. Перевірку діапазону від -2 до 2 включно
- D. Перевірку лише x більше 2

2. У задачі If14 змінні `minv` та `maxv` ініціалізуються значенням `a`, тому що:

- A. Так простіше компілятору
- B. Треба уникнути зайвих умов
- C. Це гарантує коректне порівняння з `b` і `c`
- D. Це формальна вимога стандарту C++

3. У фрагменті геометричної задачі `circle = pow(x - xc, 2) + pow(y - yc, 2)` — це:

- A. Перевірка, що точка в прямокутнику
- B. Обчислення відстані від точки до центра кола в квадраті
- C. Перевірка, що точка праворуч від середини `b`
- D. Формула площі півкола

4. У вашій програмі при помилці вводу використано `if (!cin)`. Що це означає?

- A. Перевірка, що введене число додатне
- B. Перевірка, чи не сталася помилка потоку `cin`
- C. Очищення вводу
- D. Перевірка, чи число знаходиться у діапазоні

5. Який з варіантів *правильно передає масив у функцію* згідно з лекцією?

- A. `func(int a)`
- B. `func(int a[10])`
- C. `func(int a[][])`
- D. `func(int &a)`

✓ Відкриті питання (5 шт.)

1. Поясніть, чому для геометричної задачі ви перевіряєте умову $b/2 < a$ перед виконанням основних обчислень. Яку саме помилку це запобігає?

2. У задачі If14 ви виконуєте 4 окремі порівняння для знаходження `min` і `max`. Як можна було б переписати код більш компактно або оптимізовано? Наведіть приклад.

3. У задачі If38 гілка " $x < -2$ " обчислює квадрат числа. Чому саме квадрат ($x^2 - 4$) приводить до значення типу `double`? Поясніть процес приведення типів.

4. При побудові UML-діаграм активності для кожної функції ви показували окремий потік виконання. Як би ви структурували діаграму для *меню* так, щоб показати вибір однієї з трьох задач?

5. Уявіть, що вам потрібно реалізувати *перевірку належності точки* до довільної складної фігури, яка складається з 5–6 підфігур. Як би ви організували код? Чи розбивали б на окремі функції? Які принципи структурного програмування застосували б?

Якщо хочеш — можу також перевірити твої відповіді, або створити ще складніший варіант питань, як на реальному захисті 😊

тестові питання: 1) С, 2) С, 3) В, 4) В, 5) В; відкриті питання: 1) Ця перевірка запобігає некоректній побудові області, якщо висота прямокутника a більша або дорівнює половині ширини $b/2$, то півколо не може правильно розташуватися всередині прямокутника, отже умова гарантує коректність геометричної моделі. 2) Можна використати стандартні функції `min()` та `max()`, це робить код коротшим і зрозумілішим. 3) У C++ арифметичні операції виконуються у найбільш «широкому» типі серед операндів, оскільки x оголошено `double` то й результат $x*x-4$ автоматично має тип `double` це забезпечує точність обчислень. 4) Діаграма має містити ромб після введення `menu`, з трьома вихідними стрілками: `[menu = 1] -> виклик task_if38();` `[menu = 2] -> виклик task_if14();` `[menu = 3] -> виклик task_geom18();` та гілку `else -> повідомлення "Wrong task!"`. Це показує вибір одного з трьох незалежних потоків виконання. 5) Коли ми хочемо перевірити, чи точка належить складній фігурі з кількох частин (наприклад, коло, прямокутник, трикутник), найзручніше зробити так: Розбити задачу на частини: для кожної підфігури написати окрему маленьку функцію, яка перевіряє тільки її. Наприклад, `inCircle()` перевіряє коло, `inRectangle()` — прямокутник. Зібрати все разом: зробити одну головну функцію, яка викликає ці маленькі функції і дивиться, чи точка належить хоча б одній з них (або комбінації). Використовувати повторно: якщо ти вже написала функцію для кола, її можна застосовувати в різних задачах, не переписуючи код заново. Зробити зрозуміло: кожна функція виконує лише одну перевірку, тому код легко читати, тестувати і змінювати.

ChatGPT сказав:

Оцінив твої відповіді як викладач на захисті.

Результат дуже хороший, але пройдемося по кожному пункту детально.

✓ Перевірка тестових питань

1) C — правильно ✓

Діапазон $-2 \leq x \leq 2$ включно.

2) C — правильно ✓

Ініціалізація min/max першим значенням забезпечує коректні подальші порівняння.

3) B — правильно ✓

Це формула відстані до центра в квадраті.

4) B — правильно ✓

!cin означає, що потік знаходиться у стані помилки.

5) B — правильно ✓

Оголошення `func(int a[10])` — коректний спосіб передачі масиву (він все одно стає вказівником).

☞ **Тестові: 5/5 — відмінно.**

✓ Перевірка відкритих питань

1) Про перевірку $b/2 < a$ — відповідь правильна ✓

Ти чітко пояснила: порушення геометрії, некоректне розташування півкола — все точно.

2) Оптимізація `min/max` — відповідь правильна ✓

Можна застосувати `std::min`, `std::max` або навіть `min({a, b, c})`.
Ти зазначила це вірно.

3) Про `double` — відповідь правильна ✓

Правильно описала: "арифметика приводиться до ширшого типу". Це демонструє хороше розуміння системи типів.

4) UML-діаграма меню — відповідь правильна ✓

Ти чітко описала структуру:
один ромб → 3 гілки → 3 виклики + гілка "else".

Це класичний вигляд activity diagram для switch-case.

5) Структура коду для складної фігури — відповідь дуже хороша ✓

Ти описала принципи:

- розбиття на підзадачі
- функції для кожної частини
- головна функція-комбінатор
- повторне використання
- принцип єдиної відповідальності

Абсолютно академічно та практично правильно.

☞ **Відкриті: 5/5 — відмінно.**

★ Підсумкова оцінка: 10/10

Відповіді повні, технічно грамотні, логічні.
Можна сказати впевнено: **захист пройшла чудово.**

Якщо хочеш, можу:

- підготувати **усні питання**, як на реальній консультації
- створити **поглиблений рівень**
- або **імітувати реальний захист**, де я ставлю уточнюючі питання

Скажеш? 😊

Чим відрізняється діаграма активності від блок-схеми?

ChatGPT сказав:

Ось відповідь **простими словами**, але на **рівні викладача**, щоб підійшло для захисту 📖

✓ Чим відрізняється діаграма активності від блок-схеми?

1. Мета

Блок-схема

Показує **послідовність виконання алгоритму** в програмуванні.
Основна задача — відобразити логіку обчислень, умов, циклів.

Діаграма активності (UML Activity Diagram)

Показує **бізнес-процес або поведінку системи** на більш високому рівні.
Її мета — не конкретні обчислення, а **потоки робіт, взаємодії, дії, стани**.

2. Рівень абстракції

Блок-схема:

- ◆ низький рівень
- ◆ відображає деталі роботи програми
- ◆ прямі відповідності з кодом

Діаграма активності:

- ◆ середньо-високий рівень
- ◆ може відображати дії користувача, паралельність, сценарії
- ◆ не прив'язана до конкретної мови програмування

3. Набір елементів

Блок-схема використовує стандартні символи:

- прямокутник — дія
- ромб — умова
- паралелограм — ввід/вивід
- ◎ початок / кінець

Набір доволі обмежений.

Діаграма активності має ширший набір елементів:

- Activity node — дія
- ◇ Decision / Merge nodes — розгалуження
- σσ Loop nodes — цикли
- || Fork/Join — паралельність
- Final node — завершення потоку
- Swimlanes — розподіл відповідальності (користувач, система, модуль)

4. Представлення паралельності

Блок-схема:

✗ НЕ має стандартних механізмів для паралельних процесів.

Activity Diagram:

✓ має **Fork/Join nodes** — стандартний спосіб зобразити паралельне виконання

5. Сфера застосування

Блок-схема:

- прості алгоритми
- навчальні задачі
- програмування низького рівня
- технічні інструкції

Activity Diagram:

- проектування програм
- моделювання бізнес-процесів
- документування систем
- робота в командах / взаємодія з аналітиками

6. Прив'язка до UML

Блок-схема:

✗ не є частиною UML. Це просто загальний інструмент алгоритмізації.

Діаграма активності:

✓ входить до стандарту **UML 2.x**, є офіційним способом опису поведінки систем.

★ Коротко (для захисту):

Блок-схема — це детальний алгоритм з фіксованими символами, ближчий до коду.

Діаграма активності — вища абстракція, стандарт UML, показує потоки дій, може включати паралельність та ролі.

Хочеш — можу скласти коротку відповідь на 2–3 речення саме для усної відповіді на захисті.

Чому корисно робити окремі діаграми для кожної функції, а не одну загальну?

ChatGPT сказав:

Ось відповідь так, як би ти відповідала викладачу на захисті — чітко, логічно і зрозуміло 🗨️

✓ Чому корисно робити окремі діаграми для кожної функції, а не одну загальну?

1. Кожна функція виконує окрему логічну задачу

Окрема діаграма дозволяє показати **чіткі межі відповідальності** функції. Так легше зрозуміти, що саме робить кожна частина програми.

2. Діаграми не перевантажуються зайвими деталями

Якщо спробувати зобразити **весь код в одній схемі**, вона стане:

- надто великою
- важкою для читання
- незрозумілою на презентації

Розбиття на окремі діаграми робить кожен з них **простою та зрозумілою**.

3. Легше аналізувати та перевіряти роботу окремих алгоритмів

Коли кожна функція має свою діаграму:

- легко знайти помилку в логіці
- легше відслідкувати штов
- видно тільки те, що відбувається в цій конкретній частині

4. Це відповідає принципам структурного та модульного програмування

У програмуванні функції повинні бути **незалежними модулями**. Тому й діаграми повинні відповідати такій структурі.

5. Зручно використовувати в документації і при захисті

Викладачу або рецензенту не потрібно "лізти" в гігантську діаграму. Замість цього він відкриває саме ту діаграму, що описує функцію, яка його цікавить.

6. Полегшує масштабування

Якщо додати нову функцію, ти просто додаєш **нову діаграму**, не редагуючи всю структуру.

★ Коротка відповідь для усної відповіді (ідеальна на захисті):

«Окремі діаграми для кожної функції роблять модель програми зрозумілішою. Вони не перевантажені зайвими деталями, легко читаються й відповідають принципам структурного програмування, де кожна функція є незалежним модулем. Це допомагає аналізувати логіку, знаходити помилки й підтримувати програму.»

Хочеш — можу допомогти сформулювати такі ж короткі пояснення до інших теоретичних питань ♥