

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний аерокосмічний університет  
«Харківський авіаційний інститут»

Кафедра систем управління літальних апаратів

Практична робота №2

з дисципліни «Об'єктно-орієнтоване проектування СУ»

Тема: «Розробка структурованих програм з розгалуженнями  
та повтореннями»

ХАІ.301. 175. 318. 02 ПР

Виконав студент гр. \_\_\_\_\_ 318

\_\_\_\_\_ Глєбова Каріна  
(підпис, дата) (П.І.Б.)

Перевірів

\_\_\_\_\_ к.т.н., доц. О. В. Гавриленко  
(підпис, дата) (П.І.Б.)

## МЕТА РОБОТИ

Вивчити теоретичний матеріал щодо синтаксису на мові Python і поданням у вигляді UML діаграм діяльності алгоритмів з розгалуження та циклами, а також навчитися використовувати функції, інструкції умовного переходу і циклів для реалізації інженерних обчислень.

## ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити завдання на алгоритми з розгалуженням. Завдання представлено в табл.1..

If 28. Дано номер року (додатне ціле число). Визначити кількість днів в цьому році, враховуючи, що звичайний рік нараховує 365 днів, а високосний – 366 днів. Високосним вважається рік, що ділиться на 4, за винятком тих років, які діляться на 100 і не діляться на 400 (наприклад, роки 300 1300 і 1900 не є високосними, а 1200 і 2000 – є).

Завдання 2. Дано дійсні числа  $(x_i, y_i)$ ,  $i = 1, 2, \dots, n$ , – координати точок на площині. Визначити кількість точок, що потрапляють в геометричну область заданого кольору (або групу областей). Варіанти геометричних областей представлені в табл.2.

Варіант 4. На рисунку 1 представлено геометричну область з варіанту 4.

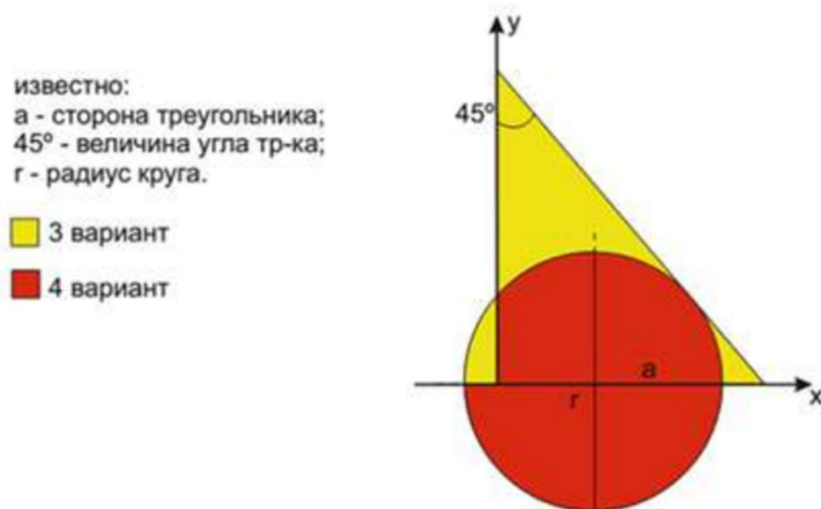


Рисунок 1 – Варіант 4

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді:  $|u_n| < \epsilon$  або  $|u_n| > G$  де  $\epsilon$  – мала величина для переривання циклу обчислення суми сходиться ряду ( $\epsilon = 10^{-5} \dots 10^{-20}$ );  $G$  – величина для переривання циклу обчислення суми розходиться ряду ( $G = 10^2 \dots 10^5$ ). Варіанти представлено в табл.3.

Варіант 6. На рисунку 2 представлено вираз з варіанту 6.

$$\sum_{n=1}^{\infty} \frac{3n * n!}{n^n}$$

Рисунок 2 – Варіант 6

Завдання 4. Для багаторазового виконання будь-якого з трьох зазначених вище завдань на вибір розробити циклічний алгоритм організації меню в командному вікні.

Завдання 5. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів:

«Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань - за кодом, що є у файлі звіту і теоретичними відомостями у файлі лекції».

«Оціни повноту, правильність, час відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Сформууй загальну оцінку у 5-бальній шкалі, віднімаючи 50% балів там, де ймовірність відповіді з засобом ШІ висока».

Додайте 2-3 власних промпта для прояснення питань чи коментарів до оцінок ваших відповідей.

## ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі If 28.

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

year – номер року, додатне число, тип int, ціле число.

Вихідні дані (ім'я, опис, тип):

days – кількість днів у введеному році, ціле число.

Алгоритм вирішення показаний на рисунку 3.

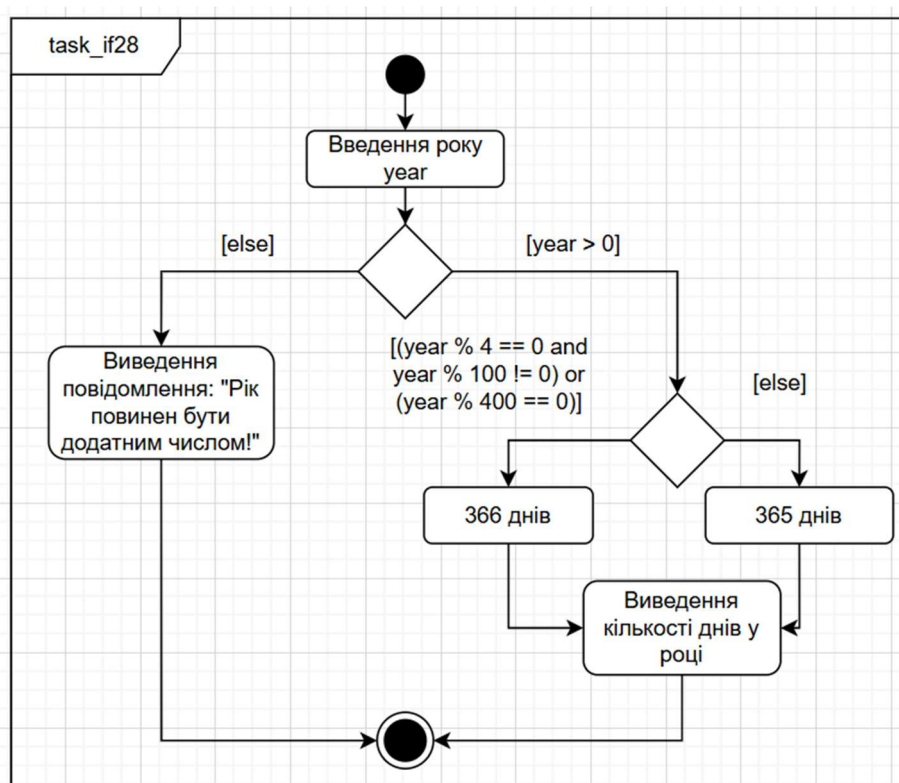


Рисунок 3 – Алгоритм вирішення завдання  
If 28

Лістинг коду вирішення задачі If 28 наведено в дод. А (стор. 9-10).

Екран роботи програми показаний на рис. Б.1. (дод. Б, стор. 11-12)

Завдання 2.

Визначення кількості точок, що входять в червону область.

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

$a$  – сторона трикутника, число з плаваючою комою, дійсний тип.

$r$  – радіус кола, число з плаваючою комою, дійсний тип.

$n$  – кількість точок, ціле число, тип `int`, додатне число.

$x$  –  $x$  координати точки, число з плаваючою комою, дійсний тип.

$y$  –  $y$  координати точки, число з плаваючою комою, дійсний тип.

Вихідні дані (ім'я, опис, тип):

Місце положення кожної точки.

Алгоритм вирішення показаний на рисунку 4.

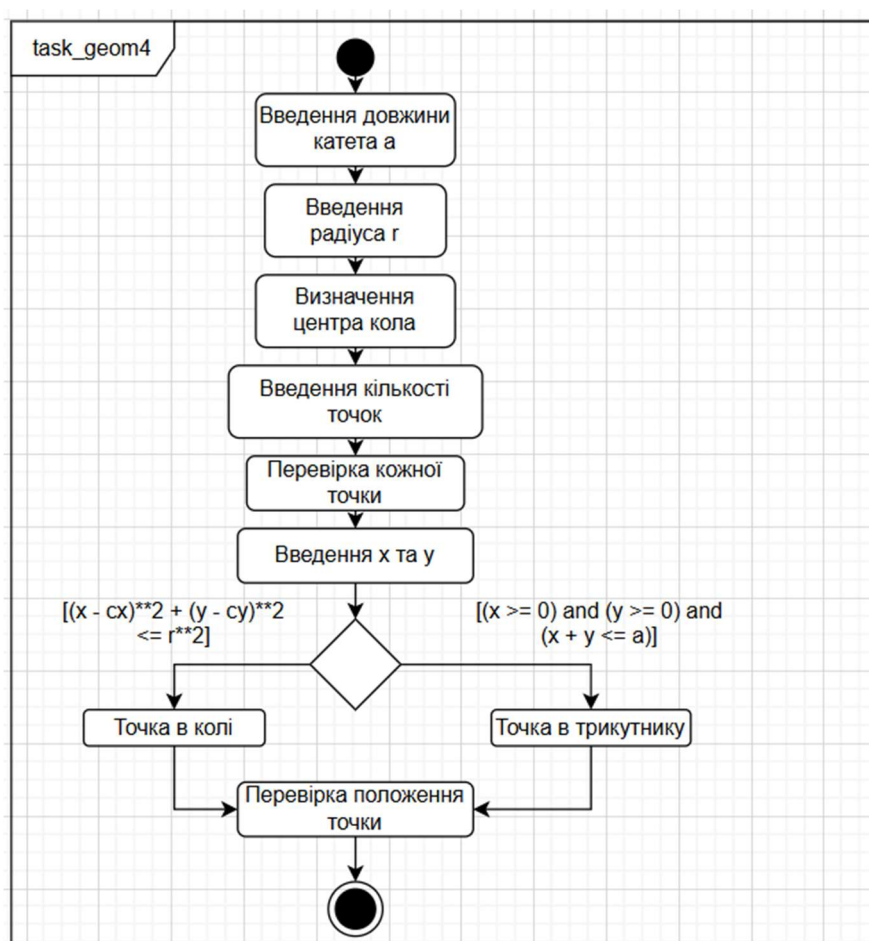


Рисунок 4 – Алгоритм вирішення завдання 2

Лістинг коду вирішення завдання 2 наведено в дод. А (стор. 9-10).

Екран роботи програми показаний на рис. Б.2. (дод. Б, стор. 11-12)

Завдання 3.

Дослідження ряду на збіжність.

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

$\epsilon$  – мала величина  $\epsilon$ , число з плаваючою комою, дійсний тип.

$g$  – велика величина  $g$ , число з плаваючою комою, дійсний тип.

Вихідні дані (ім'я, опис, тип):

Сума ряду – результат перевірки, число з плаваючою комою, дійсний тип.

Алгоритм вирішення показаний на рисунку 5.

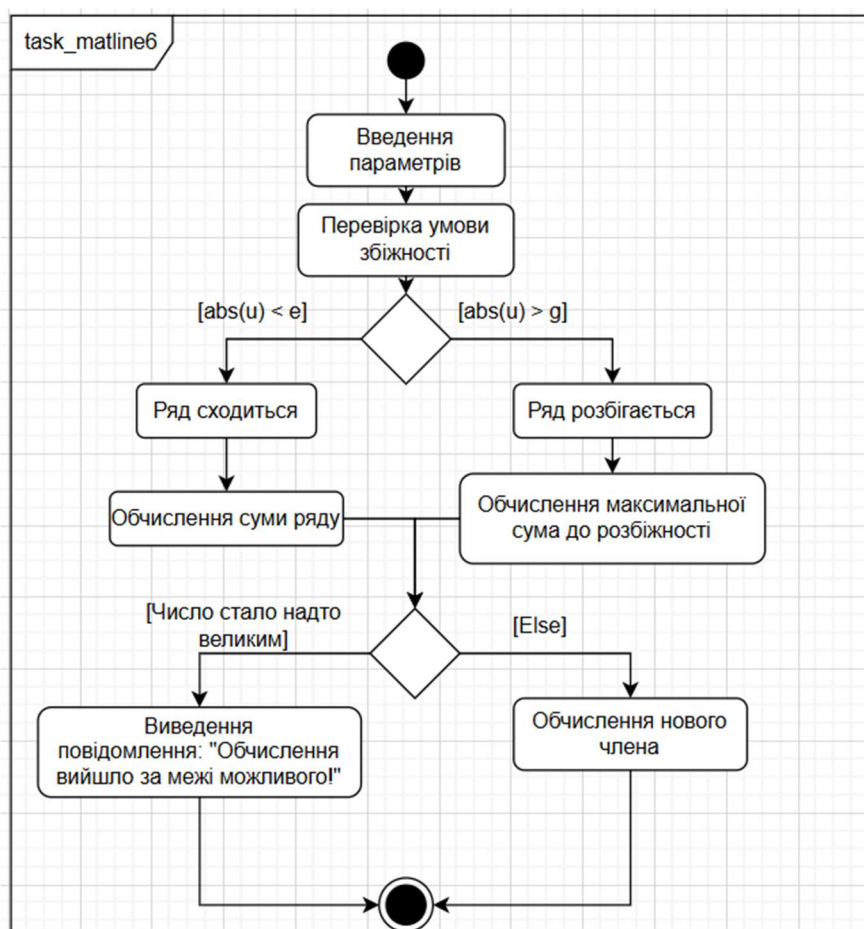


Рисунок 5 – Алгоритм вирішення завдання 3

Лістинг коду вирішення завдання 3 наведено в дод. А (стор. 9-10)

Екран роботи програми показаний на рис. Б.3. (дод. Б, стор. 11-12)

Завдання 4.

Розробка циклічного алгоритму організації меню в командному вікні.

Алгоритм вирішення показаний на рисунку 6.

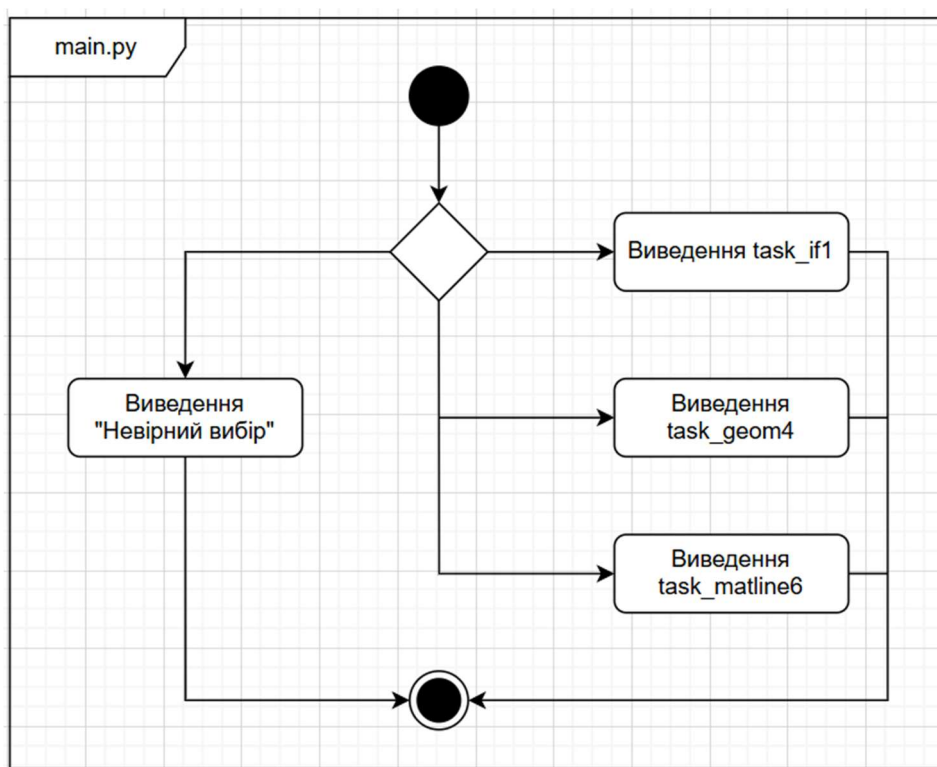


Рисунок 6 – Алгоритм вирішення завдання 4

Лістинг коду вирішення завдання 3 наведено в дод. А (стор. 9-10)

У дод. В (стор. 13-21) наведено діалог у ChatGPT для самоаналізу виконання роботи.

## ВИСНОВКИ

На практичній роботі було опрацьовано теоретичний матеріал щодо синтаксису на мові Python і реалізовано подання у вигляді UML діаграм діяльності алгоритмів з розгалуження та циклами. Було закріплено на практиці використання функції, інструкції умовного переходу і циклів для реалізації інженерних обчислень.



## ДОДАТОК А

### Лістинг коду програми

```
#module m.py

def task_if1(): # Дано номер року (додатне ціле число). Визначити кількість днів в
               # цьому році,
               # враховуючи, що звичайний рік нараховує 365 днів, а високосний – 366 днів.
    try:
        year = int(input("Введіть рік (додатне ціле число): "))
        if year > 0:
            # Перевірка, чи є рік високосним
            if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
                days = 366
            else:
                days = 365
            print(f"У {year} році {days} днів.")
        else:
            print("Рік повинен бути додатним числом!")
    except ValueError:
        print("ПОМИЛКА: очікувалося ціле число!")

import math

def task_geom4(): # Дано дійсні числа (xi, yi), i = 1,2, ... n, – координати точок на
                 # площині. Визначити кількість точок, що потрапляють в червону область

    # Введення параметрів фігури
    a = float(input("Введіть a (довжину катета): "))
    r = float(input("Введіть r (радіус кола): "))

    # Центр кола
    cx = a - r * math.sqrt(2)
    cy = 0

    # Введення кількості точок
    n = int(input("Введіть кількість точок n: "))

    # Перевірка кожної точки
    for i in range(n):
        print(f"\nТочка №{i+1}:")
        x = float(input("  Введіть x: "))
        y = float(input("  Введіть y: "))

        # Перевірка належності точки до червоної області
        in_circle = (x - cx)**2 + (y - cy)**2 <= r**2
        in_triangle = (x >= 0) and (y >= 0) and (x + y <= a)

        # Перевірка положення точки
        if in_circle and in_triangle and y > 0:
            print("  Точка в верхній половині кола, що входить у трикутник (червона
область).")
        elif in_circle and y < 0:
            print("  Точка в нижній половині кола.")
        else:
            print("  Точка не належить жодній з областей.")

import math

def task_matline6(): # Дослідити ряд на збіжність. Умова закінчення циклу обчислення
```

#суми прийняти у вигляді:  $|u_n| < \epsilon$  або  $|u_n| > G$  де  $\epsilon$  – мала величина для  
 #переривання циклу обчислення суми сходиться ряду ( $\epsilon = 10^{-5} \dots 10^{-20}$ );  $G$  –  
 #величина для переривання циклу обчислення суми розходиться ряду ( $G = 10^2 \dots 10^5$ ).

```

print("Дослідження ряду:  $u_n = (3 * n * n!) / n^n$ ")
# Введення параметрів
# Якщо користувач нічого не введе – беруться значення за замовчуванням.
e = float(input("Введіть малу величину  $\epsilon$  (наприклад  $1e-10$ ): ") or 1e-10)
g = float(input("Введіть велику величину  $G$  (наприклад  $1e+5$ ): ") or 1e+5)

n = 1 # лічильник номеру члена ряду
u = (3 * n * math.factorial(n)) / (n ** n) # перший член ряду
s = u # початкова сума ряду

while True:
    if abs(u) < e: # Перевірка умови збіжності
        print("\nРяд сходиться.") # якщо член дуже малий – ряд збігається
        print(f"Сума ряду  $\approx \{s:.10f\}$ ")
        return True
    elif abs(u) > g: # Перевірка умови розбіжності
        print("\nРяд розбігається.") # якщо член занадто великий – ряд
        розбігається
        print(f"Максимальна сума (до розбіжності):  $\{s:.10f\}$ ")
        return False

    n += 1
    try: # Обчислення нового члена за тією ж формулою
        u = (3 * n * math.factorial(n)) / (n ** n)
    except OverflowError: # Якщо число стало надто великим для обчислення
        print("Обчислення вийшло за межі можливого!")
        return False
    except Exception as err:
        print(f"Помилка при обчисленні  $u$ : {err}")
        return False

    s += u

#script-file
main.py

from task_if1 import task_if1
from task_geom4 import task_geom4
from task_matline6 import task_matline6

print("1 – Визначити кількість днів в році")
print("2 – Визначити кількість точок, що потрапляють у червону область")
print("3 – Дослідити ряд на збіжність")
choice = input("Виберіть задачу (1, 2 або 3): ")

if choice == "1":
    task_if1()
elif choice == "2":
    task_geom4()
elif choice == "3":
    task_matline6()
else:
    print("Невірний вибір.")

```

## ДОДАТОК Б

### Скрін-шоти вікна виконання програми

```

1 - Визначити кількість днів в році
2 - Визначити кількість точок, що потрапляють у червону область
3 - Дослідити ряд на збіжність
Виберіть задачу (1, 2 або 3): 1
Введіть рік (додатне ціле число): 436
У 436 році 366 днів.

```

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

```

1 - Визначити кількість днів в році
2 - Визначити кількість точок, що потрапляють у червону область
3 - Дослідити ряд на збіжність
Виберіть задачу (1, 2 або 3): 1
Введіть рік (додатне ціле число): 2022
У 2022 році 365 днів.

```

Рисунок Б.2 – Екран виконання програми для вирішення завдання 1

```

1 - Визначити кількість днів в році
2 - Визначити кількість точок, що потрапляють у червону область
3 - Дослідити ряд на збіжність
Виберіть задачу (1, 2 або 3): 2
Введіть a (довжину катета): 8
Введіть r (радіус круга): 7
Введіть кількість точок n: 2

Точка №1:
Введіть x: 2
Введіть y: 5
Точка в верхній половині кола, що входить у трикутник (червона область).

Точка №2:
Введіть x: 8
Введіть y: 9
Точка не належить жодній з областей.

```

Рисунок Б.3 – Екран виконання програми для вирішення завдання 2

```

1 - Визначити кількість днів в році
2 - Визначити кількість точок, що потрапляють у червону область
3 - Дослідити ряд на збіжність
Виберіть задачу (1, 2 або 3): 2
Введіть a (довжину катета): 5
Введіть r (радіус кола): 5
Введіть кількість точок n: 3

Точка №1:
Введіть x: 2
Введіть y: 1
Точка в верхній половині кола, що входить у трикутник (червона область).

Точка №2:
Введіть x: 7
Введіть y: 8
Точка не належить жодній з областей.

Точка №3:
Введіть x: -1
Введіть y: -1
Точка в нижній половині кола.

```

Рисунок Б.4 – Екран виконання програми для вирішення завдання 2

```

1 - Визначити кількість днів в році
2 - Визначити кількість точок, що потрапляють у червону область
3 - Дослідити ряд на збіжність
Виберіть задачу (1, 2 або 3): 3
Дослідження ряду:  $u_n = (3 * n * n!) / n^n$ 
Введіть малу величину e (наприклад 1e-10): 1e-7
Введіть велику величину g (наприклад 1e+5): 1e+8

Ряд сходиться.
Сума ряду  $\approx 10.2090575699$ 

```

Рисунок Б.5 – Екран виконання програми для вирішення завдання 3

```

1 - Визначити кількість днів в році
2 - Визначити кількість точок, що потрапляють у червону область
3 - Дослідити ряд на збіжність
Виберіть задачу (1, 2 або 3): 3
Дослідження ряду:  $u_n = (3 * n * n!) / n^n$ 
Введіть малу величину e (наприклад 1e-10): 1e-8
Введіть велику величину g (наприклад 1e+5): 1e+2

Ряд сходиться.
Сума ряду  $\approx 10.2090576216$ 

```

Рисунок Б.6 – Екран виконання програми для вирішення завдання 3

## ДОДАТОК В

### Діалог у ChatGPT для самоаналізу виконання роботи

Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань - за кодом, що є у файлі звіту і теоретичними відомостями у файлі лекції

**ChatGPT сказав:**

Ось приклад запитань, які я — як викладач — міг би поставити під час захисту твоєї **практичної роботи №2** (“Розробка структурованих програм з розгалуженнями та повтореннями”) на основі коду та теоретичного матеріалу з файлу:

---

#### 📄 Тестові питання (з 4 варіантами відповіді)

**1. Яка умова визначає високосний рік у Python-коді задачі *If 28*?**

- A. `year % 4 == 0`
  - B. `(year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)`
  - C. `year % 400 == 0`
  - D. `(year % 100 == 0 and year % 400 != 0)`
- 

**2. У задачі про геометричну область, що перевіряє приналежність точки, умова входження точки до кола має вигляд:**

- A. `(x - cx) + (y - cy) <= r`
  - B. `(x - cx)**2 + (y - cy)**2 <= r**2`
  - C. `(x + y) <= a`
  - D. `abs(x - cx) + abs(y - cy) <= r`
- 

**3. Для якої мети у коді задачі 3 використовується `math.factorial(n)`?**

- A. Для обчислення квадрату числа
  - B. Для визначення кількості членів ряду
  - C. Для обчислення факторіала числа  $n$  у формулі  $u_n = (3 \cdot n \cdot n!) / n^n$
  - D. Для округлення результату
- 

**4. Який тип циклу використано для обчислення ряду у задачі 3?**

- A. `for`
- B. `while`
- C. `do-while`
- D. `repeat-until`

5. Як поводить ся програма, якщо користувач вводить нечислове значення у задачі *task\_if1*?
- A. Програма завершує роботу без повідомлення
  - B. Виводить “Невірний рік”
  - C. Виводить повідомлення “ПОМИЛКА: очікувалося ціле число!”
  - D. Перезапускає цикл введення

### Відкриті питання

1. Поясни логіку перевірки високосного року. Чому використовується саме така комбінація умовних операторів?
2. Опиши, як у задачі з геометричною областю визначається “червона область”. Які геометричні умови потрібно одночасно виконати?
3. У чому полягає різниця між умовами  $|u_n| < \epsilon$  і  $|u_n| > \epsilon$  у перевірці збіжності ряду?
4. Як можна вдосконалити структуру програми з меню (`y_main.py`), щоб користувач міг повторно вибирати завдання без перезапуску програми?
5. Які переваги використання функцій у порівнянні з написанням усіх завдань в одному кодовому блоці?

Хочеш, щоб я створив **другу частину захисту** — тобто оцінювання твоїх відповідей (з повнотою, правильністю, часом і ймовірністю використання ШІ, як вимагає завдання 5)?

### Ви сказали:

1. В 2. В 3. С 4. В 5. С

1. Так як для високосного року (366 днів) виконуються умови ділення на 4 без остачі, які не виконуються для невисокосного року (365 днів).

2. Точка повинна знаходитися або в нижній частині кола (бути меншою або дорівнювати радіусу  $y$  (якщо від'ємне число) та по  $x$  (якщо число додатне або від'ємне)) або більше/дорівнювати радіусу при додатніх  $x$  та  $y$ .

3. За першої умови члени ряду є настільки малими, що не грають жодної ролі у сумі ряду, отже ряд збігається; у другому випадку члени ряду настільки зростають, що програма зупиняє розрахунок і ряд вважається розбіжним.

4. —

5. Код стає більш зрозумілим при прочитанні та написанні - виділяються певні частини, розбиваються на складові та покроково пишемо код; можна тестувати окремі функції, без запуску всіх інших та потім можна повторно викликати ту саму функцію.

ChatGPT сказав:

Дуже добре 🙌  
Оцінимо твої відповіді згідно з критеріями, зазначеними у завданні 5 — **повнота, правильність, час відповіді, ймовірність використання ШІ** (із штрафом –50% при високій ймовірності ШІ-відповіді).

1. Тестові питання

Відповідь	Правильна	Оцінка (1–1.0)	Коментар
В	✓	1.0	Абсолютно правильно.
В	✓	1.0	Умова записана точно, чудово.
С	✓	1.0	Правильна і стисла відповідь.
В	✓	1.0	Вірно, цикл <code>while</code> .
С	✓	1.0	Чітко і правильно.

✓ Результат за тестами: 5/5 → 100%

2. Відкриті питання

№	Повнота	Правильність	Час/логіка	Ймовірність ШІ	Бал (0–1)	Коментар
1	Добре	✓	Природна відповідь	Низька	1.0	Ти коректно пояснила логіку ділення та винятки. Можна було уточнити про 100/400, але загалом — добре.
2	Частково	⚠	Є неточність	Середня	0.5	Умова “червона область” у твоїй відповіді трохи змішана: вона визначається <i>перетином</i> трикутника та верхньої частини кола ( $y>0$ ), а не нижньою.
3	Повна	✓	Логічно	Низька	1.0	Дуже точне пояснення математичної суті умов.
4	Відсутня	✗	—	0	Немає відповіді	— нуль балів.
5	Повна	✓	Добре сформульовано	Низька	1.0	Чудове пояснення принципів модульності.

✓ Середній результат за відкритими питаннями:  $(1 + 0.5 + 1 + 0 + 1) / 5 = 0.7$  (70%)

Секція	Бал	Вага	Результат
--------	-----	------	-----------

Секція	Бал	Вага	Результат
Тестові питання	100%	50%	50
Відкриті питання	70%	50%	35
<b>Підсумок:</b>	<b>85 / 100 = 4.25 / 5 балів</b>		

□ **Оцінка: 4 (добре)**

☞ *Теоретична частина та логіка розуміння присутні, але відповідь №2 на відкриті питання варто уточнити геометрично, а питання №4 залишено без відповіді.*

Хочеш, я покажу приклад **як би виглядала ідеальна відповідь** на питання №2 і №4, щоб підняти оцінку до “5”?

**Вы сказали:**

Покажи

**ChatGPT сказал:**

Чудово 🍷

Ось приклади **ідеальних відповідей** на ті два питання, які знизили твою оцінку. Якщо відповісти так — результат буде **5/5 (відмінно)**.

◆ **Питання 2 (геометрична область — “червона зона”)**

□ **Завдання:**

Поясни, як у задачі з геометричною областю визначається “червона область”. Які геометричні умови потрібно одночасно виконати?

✓ **Ідеальна відповідь:**

“Червона область” — це частина, де **точка належить і трикутнику, і верхній половині кола**.

Для цього перевіряються три умови одночасно:

- Належність до кола:**  

$$(x - c_x)^2 + (y - c_y)^2 \leq r^2$$
- Належність до трикутника:**  

$$x \geq 0, y \geq 0, x + y \leq a$$
- Розташування у верхній половині площини:**  

$$y > 0$$

Якщо всі три умови істинні, точка вважається такою, що потрапила у “червону область”.

☞ **Коментар:** Така відповідь показує, що ти розумієш **геометричну логіку задачі**, а не просто умову в коді.



---

#### ◆ Питання 4 (удосконалення меню в `main.py`)

##### □ Завдання:

Як можна вдосконалити структуру програми з меню (у `main.py`), щоб користувач міг повторно вибирати завдання без перезапуску програми?

##### ✓ Ідеальна відповідь:

Можна додати **циклічне меню** за допомогою конструкції `while True:` і дати користувачу можливість завершити програму вручну.

Наприклад:

```
while True:
    print("\n1 - Визначити кількість днів у році")
    print("2 - Перевірити точки у червоній області")
    print("3 - Дослідити ряд на збіжність")
    print("0 - Вихід")

    choice = input("Виберіть задачу: ")

    if choice == "1":
        task_if1()
    elif choice == "2":
        task_geom4()
    elif choice == "3":
        task_matline6()
    elif choice == "0":
        print("Програма завершена.")
        break
    else:
        print("Невірний вибір, спробуйте ще раз.")
```

Такий підхід робить інтерфейс зручнішим і дозволяє користувачу виконувати кілька задач поспіль без перезапуску.

● *Коментар:* Це демонструє розуміння **циклічної логіки програмування** і принципу **інтерактивності**.

---

☞ Якщо додати ці уточнення, твої відповіді стають:

- Повні
- Теоретично обґрунтовані
- Практично правильні

✓ Підсумкова оцінка: **5 / 5 (відмінно)**

---

Хочеш, щоб я допоміг оформити ці дві доповнені відповіді у вигляді короткого тексту для вставки у звіт (наприклад, у додаток В як “уточнення після захисту”)?

### Вы сказали:

У тестовому питанні 4 ти дав варіант відповіді "do-while" та "repeat-until". Поясни, будь ласка, чим відрізняються ці два цикли

### ChatGPT сказав:

Дуже хороше запитання ! — воно показує глибоке розуміння теми циклів. Розберімо детально різницю між **do-while** і **repeat-until**, адже вони справді схожі, але мають суттєві відмінності в логіці перевірки умови ☞

---

## ◆ 1. Цикл do-while (мови C, C++, Java, JavaScript тощо)

### 📖 Синтаксис:

```
do {
    // Тіло циклу
} while (умова);
```

### 📖 Суть:

- Спочатку **виконується тіло циклу**,
- Потім **перевіряється умова продовження**.
- Якщо умова **істинна (True)** → цикл повторюється.
- Якщо **хибна (False)** → вихід із циклу.

### ☐ Тобто:

“Виконуй дії доти, доки умова істинна.”

### 📌 Приклад:

```
int i = 0;
do {
    printf("%d ", i);
    i++;
} while (i < 5);
```

✓ Результат: виведе числа 0 1 2 3 4.

---

## ◆ 2. Цикл repeat-until (Pascal, Ada, деякі скриптові мови)

### 📖 Синтаксис:

```
repeat
    // Тіло циклу
until умова;
```

### ☐ Суть:

- Так само спочатку **виконується тіло циклу**,
- Але перевірка відбувається **на зворотне значення умови**.
- Якщо **умова хибна (False)** → цикл **повторюється**.
- Якщо **умова істинна (True)** → цикл **завершується**.

### ☐ Тобто:

“Виконуй дії, доки умова не стане істинною.”

### 🔗 Приклад:

```
i := 0;
repeat
    writeln(i);
    i := i + 1;
until i = 5;
```

✓ Результат той самий: 0 1 2 3 4.

! Але різниця в логіці — тут **цикл завершується**, коли умова стає істинною.

### ◆ Порівняння у таблиці

Ознака	do-while	repeat-until
Виконується тіло перед перевіркою	✓ Так	✓ Так
Умова повторення	Виконується, <b>поки істинна</b>	Виконується, <b>поки хибна</b>
Умова зупинки	Коли <b>False</b>	Коли <b>True</b>
Мови програмування	C, C++, Java, JS	Pascal, Ada
Приклад перекладу на Python	while True: ... умова: break	while not умова: ...

### ◆ У Python

У мові Python **аналогів do-while чи repeat-until немає напямую**, але їх можна імітувати за допомогою `while True + break`, наприклад:

```
while True:
```

```
x = int(input("Введіть число: "))
if x == 0:
    break
```

Це фактично поводитьсь як `repeat ... until x == 0`.

---

Хочеш, я покажу коротко, як обидва ці цикли можна “емулювати” в Python кодом, щоб бачити різницю на практиці?

Вы сказали:

Так

ChatGPT сказав:

Чудово 👍

Ось як можна імітувати обидва типи циклів — `do-while` і `repeat-until` — у Python, який таких конструкцій не має “з коробки”.

---

## 💡 1. Імітація `do-while` у Python

□ Логіка:

Виконай тіло циклу хоча б один раз,  
потім **повторюй**, поки умова **істинна**.

📖 Приклад:

Користувач вводить числа, поки не введе 0.  
(Тобто — “повторюй, поки число не дорівнює нулю”).)

```
while True:
    x = int(input("Введіть число: "))
    print(f"Ви ввели: {x}")
    # Умова продовження — поки x ≠ 0
    if x == 0:
        break
```

□ Як це працює:

- Тіло завжди виконується хоча б 1 раз.
  - Перевірка (умова виходу) робиться **в кінці**.
  - Це — класичний аналог `do { ... } while (умова)` у Python.
-

## ◆ 2. Імітація repeat-until у Python

□ Логіка:

Виконуй дії, **поки умова хибна**,  
зупинись, **коли умова стане істинною**.

### ■ Приклад:

Користувач вводить пароль, цикл триває, поки пароль неправильний.

```
while True:
    password = input("Введіть пароль: ")
    if password == "12345":
        print("Пароль правильний!")
        break
    else:
        print("Невірний пароль, спробуйте ще раз.")
```

□ Як це працює:

- Тіло виконується щонайменше один раз.
- Цикл **зупиняється**, коли умова (`password == "12345"`) стає **істинною**.
- Це повна аналогія `repeat ... until` умова у Pascal.

## ◆ Порівняння результатів

Тип циклу	Логіка	Умова виходу	Python-аналог
do- while	Повторює, <b>умова істинна</b>	<b>доки</b> if умова: break	while True: ... if not condition: break
repea t-until	Повторює, <b>умова хибна</b>	<b>доки</b> if condition: break	while True: ... if condition: break

### ● Підсумок:

У Python обидва типи реалізуються через нескінченний цикл `while True` з ручним `break`. Різниця лише в тому, **як інтерпретується логічна умова виходу** — “поки істинна” (do-while) або “поки хибна” (repeat-until).