МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Лабораторна робота № 2

з дисципліни «Об’єктно-орієнтоване проєктування систем управлінн»

Тема: "Розробка структурованих програм з розгалуженням та повтореннями"

ХАІ.301 . Інженерія мобільних додатків. 312ст.1 ЛР

Виконав студент гр. _____312ст_____

_____Васильєв Б.А.___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
_____ ас. В. О. Білозерський
(підпис, дата) (П.І.Б.)

2024

МЕТА РОБОТИ

Вивчити теоретичний матеріал щодо синтаксису на мові Python і поданняму вигляді UML діаграм діяльності алгоритмів з розгалуження та циклами, атакож навчитися використовувати функції, інструкції умовного переходу іциклів для реалізації інженерних обчислень.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити завдання на алгоритми з розгалуженням.

If20. [На числової осі розташовані три точки: A, B, C. Визначити, яка з двох останніх точок (B або C) розташована ближче до A, і вивести цю точку і її відстань від точки A. Варіант 20

Завдання 2. Дано дійсні числа (xi, yi), i = 1,2, ... n, – координати точок на площині. Визначити кількість точок, що потрапляють в геометричну область заданого кольору (або групу областей). Варіант 11

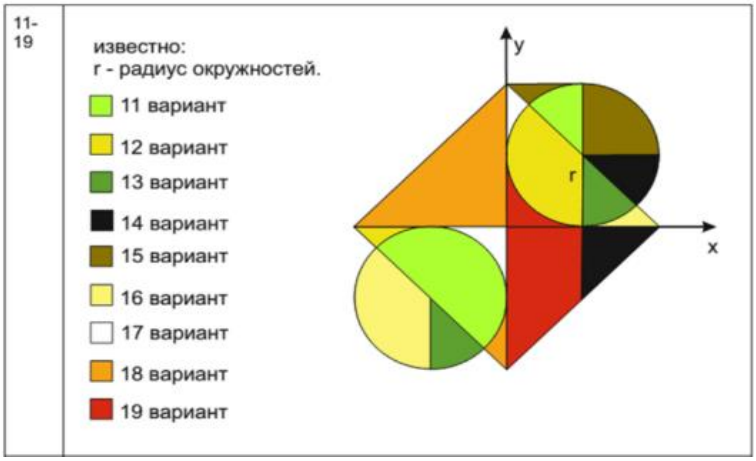

Рис.1 Завдання 2

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: | un | <E або | un | > G де e – мала величина для переривання циклу обчислення суми сходиться ряду (e = 10-5... 10-20); g – величина для переривання циклу обчислення суми розходиться ряду (g = 102...105). Варіанти представлено в табл.3. Варіант 10

10	$\sum_{n=1}^{\infty} \frac{n!-3^n}{n^n}$

Рис.2 Завдання 3

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

ВИКОНАННЯ РОБОТИ

Завдання 1: Визначення останньої та середньої цифри тризначного числа.

Вхідні дані: Тризначне ціле число, яке лежить в діапазоні від 100 до 999.

Вихідні дані: Остання та середня цифри цього числа.

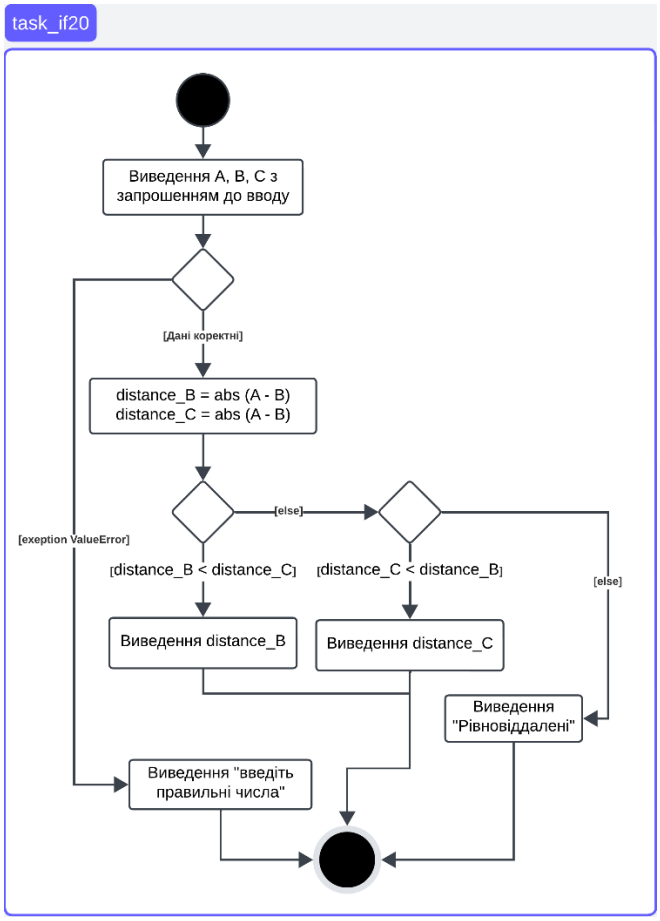

Рис. 3 діаграма Завдання 1: Визначення найближчої точки до A

Завдання 2: Визначення кількості точок в зеленій області (варіант 11)

Вхідні дані:

Дійсні числа r, x, y, де r — радіус (позитивне число), а x і y — координати точки.

Результат:

Строка, що вказує, чи знаходиться точка в зеленій області.

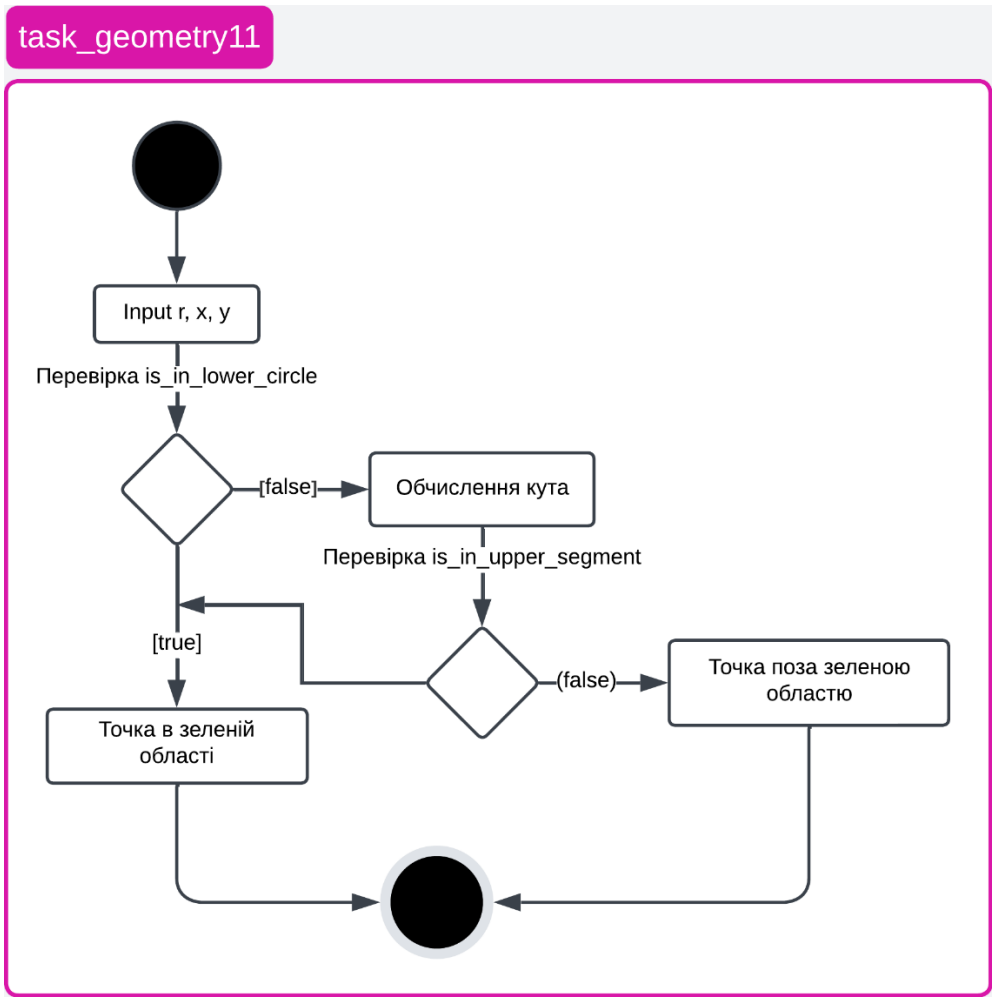

Рис. 4 діаграма Завдання 2: Визначення кількості точок в зеленій області

Завдання 3: Дослідження збіжності ряду

Вхідні дані:

Дійсне число epsilon — точність (за замовчуванням 1e-10, число).

Ціле число max_iter — максимальна кількість ітерацій (за замовчуванням 100, число).

Результат:

Строка, що вказує на результат збіжності ряду:

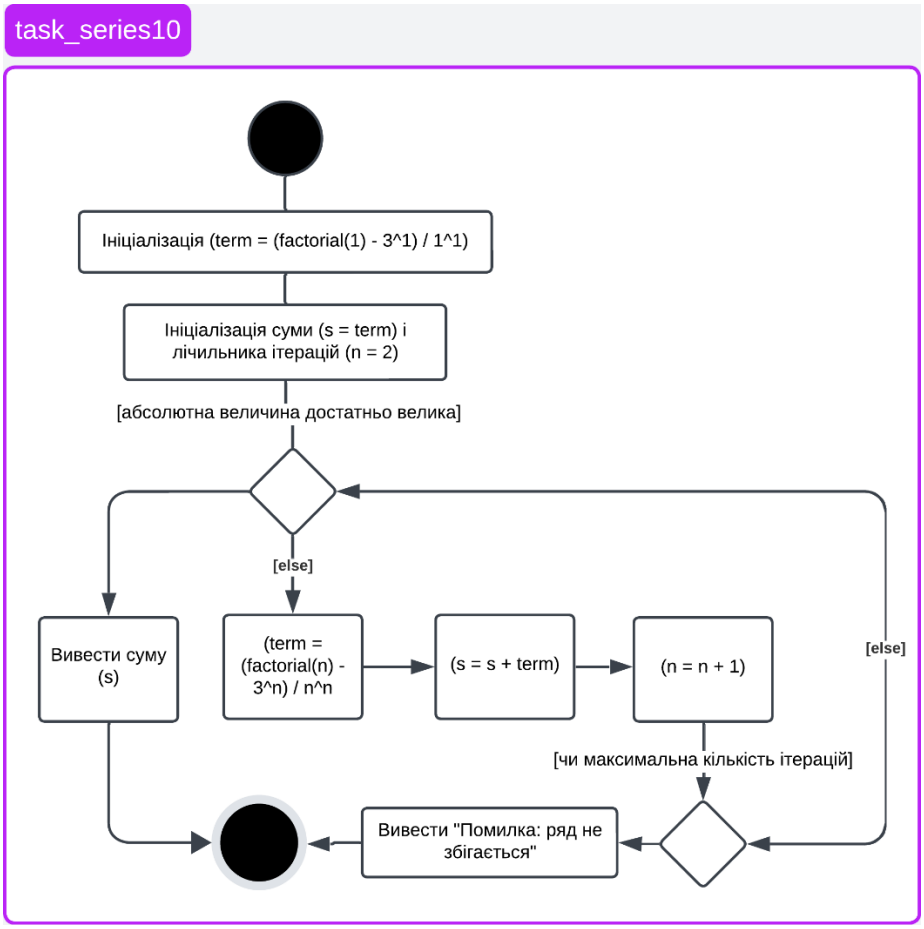

Рис. 4 діаграма Завдання 3: Дослідження збіжності ряду

ДОДАТОК А

Лістинг коду програми до завдання 1-3

```
import math as mt


# Завдання 1: Визначення найближчої точки до A
def task_if20():
    # Визначає, яка з точок B або C знаходиться ближче до точки A.

    try:
        A = float(input("Введіть координату точки A: "))
        B = float(input("Введіть координату точки B: "))
        C = float(input("Введіть координату точки C: "))

        distance_B = abs(A - B)
        distance_C = abs(A - C)

        if distance_B < distance_C:
            print(f"Точка B ближча до точки A. Відстань: {distance_B}")
        elif distance_C < distance_B:
            print(f"Точка C ближча до точки A. Відстань: {distance_C}")
        else:
            print(f"Точки B і C однаково віддалені від точки A. Відстань: {distance_B}")
    except ValueError:
        print("Введіть правильні числа.")


# Завдання 2: Визначення кількості точок в зеленій області (варіант 11)
def task_geometry11(r, x, y):
    # Умова для нижнього зеленого півкола
    is_in_lower_circle = (x ** 2 + (y + r) ** 2) <= r ** 2 and y <= 0

    # Умова для верхнього зеленого сегмента
    angle = mt.atan2(y - r, x - r)
```

```
        is_in_upper_segment = (x - r) ** 2 + (y - r) ** 2 <= r ** 2 and (0 <= angle <=
mt.pi / 2)

        # Перевірка потрапляння в зелені області
        if is_in_lower_circle or is_in_upper_segment:
            return "Точка в зеленій області!"
        else:
            return "Точка поза зеленою областю."


    # Завдання 3: Дослідження збіжності ряду
    def task_series10(epsilon=1e-10, max_iter=100):
        # Початкові параметри
        term = (mt.factorial(1) - 3 ** 1) / (1 ** 1)  # Перший член ряду
        print(f"Initial term: {term}")

        s = term  # Початкова сума
        n = 2  # Починаємо з другого члена
        e = epsilon  # Точність
        while abs(term) > e:  # Поки не досягнемо потрібної точності
            term = (mt.factorial(n) - 3 ** n) / (n ** n)
            print(f"Term at n={n}: {term}")
            s += term
            n += 1
            if n > max_iter:
                print("Error: Series did not converge within iteration limit.")
                return None

        print(f"Series converged to: {s}")
        return s
```

ДОДАТОК Б

Лістинг коду програми до завдання 4

```
import sys
from lab_2_oop import task_if20, task_geometry11, task_series10

# Завдання 4: Меню для вибору завдань
def menu():
    while True:
        print("\nВиберіть завдання:")
        print("1. Визначити, яка точка ближче до А.")
        print("2. Порахувати кількість точок у зеленій області.")
        print("3. Дослідити збіжність ряду.")
        print("0. Вихід")

        try:
            choice = int(input("Ваш вибір (0-3): "))

            if choice == 1:
                task_if20()

            elif choice == 2:
                # Отримати радіус кола
                r = float(input("Введіть радіус r: "))

                # Отримати кількість точок
                points = [(float(input("x: ")), float(input("y: "))) for _ in range(int(input("Кількість точок: ")))]

                # Лічильник для точок, що знаходяться в зеленій області
                count = 0

                # Перевірка кожної точки на належність зеленій області
                for x, y in points:
                    if task_geometry11(r, x, y) == "Точка в зеленій області!":
                        count += 1
```

```
            print(f"Кількість точок у зеленій області: {count}")

        elif choice == 3:
            task_series10()

        elif choice == 0:
            sys.exit(0)

        else:
            print("Невірний вибір. Спробуйте ще раз.")

    except ValueError:
        print("Помилка введення. Введіть число від 0 до 3.")


# Викликати головне меню:
if __name__ == "__main__":
    menu()
```

ВИСНОВКИ

У результаті виконання завдань було реалізовано три функції, які визначають найближчу точку до A, підраховують кількість точок у геометричній області та досліджують збіжність математичного ряду.

ДОДАТОК Б

Скрін-шоти вікна виконання програми

```
Введіть координату точки A: 5
Введіть координату точки B: 3
Введіть координату точки C: 8
Точка B ближча до точки A. Відстань: 2.0
```

Рисунок Б.5 – Екран виконання програми для вирішення завдання 1

```
Введіть радіус r: 5
Кількість точок: 2
x: 0
y: -3
x: 3
y: 3
Кількість точок у зеленій області: 1
```

Рисунок Б.6 – Екран виконання програми для вирішення завдання 2

```
Term at n=17: 4.299685536881045e-07
Term at n=18: 1.627181140920259e-07
Term at n=19: 6.148599408550179e-08
Term at n=20: 2.320196156205993e-08
Term at n=21: 8.744575305506187e-09
Term at n=22: 3.2920590328574556e-09
Term at n=23: 1.2380956566193408e-09
Term at n=24: 4.651958899971866e-10
Term at n=25: 1.7464069942801776e-10
Term at n=26: 6.551066071283691e-11
Series converged to: -4.783041449365289
```

Рисунок Б.7 – Екран виконання програми для вирішення завдання 3