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 програмування СУ»

Тема: "Розробка структурованих програм з розгалуженням та повтореннями"

ХАІ.301 . Авіоніка . група 3-92АВ(і). ЛР2

Виконав студент гр.\_\_\_\_\_\_\_3-92АВ(і)\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сидоренко В.В\_\_\_

(підпис, дата) (П.І.Б.)

Перевірив

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. О. В. Гавриленко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ас.  В. О. Білозерський

(підпис, дата) (П.І.Б.)

2023

# 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.

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). Варіанти геометричних областей представлені в табл.2.

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу

обчислення суми прийняти у вигляді: | un | <E або | un | > G де е – мала величина

для переривання циклу обчислення суми сходиться ряду (е = 10-5

... 10-20); g –величина для переривання циклу обчислення суми розходиться ряду (g = 102...105). Варіанти представлено в табл.3.

Завдання 4. Для багаторазового виконання будь-якого з трьох зазначених

вище завдань на вибір розробити циклічний алгоритм організації меню в

командному вікні.

Завдання 1-3 повинні бути реалізовані в окремому модулі з трьома

функціями з рядком документації, завдання 4 – у вигляді окремого скрипта з

викликом функцій з підключеного модуля.

При введенні даних і обчисленнях необхідно передбачити перевірку даних і

обробку виняткових ситуацій.

Код повинен містити коментарі!

# ВИКОНАННЯ РОБОТИ

Завдання 1. Вирішення задачі if16

* Вхідні дані:
* A (тип даних: float) - значення першої змінної
  + B (тип даних: float) - значення другої змінної
  + C (тип даних: float) - значення третьої змінної
* Вихідні дані:
  + Нові значення A, B, C залежно від умови:
    - Якщо A < B < C, то A, B, C подвоюються
    - В іншому випадку, A, B, C стають протилежними за знаком

Діапазони допустимих значень для вхідних даних (A, B, C) - будь-які дійсні числа, оскільки вони оголошені як тип даних float. Функція task\_1 виконує операції порівняння і арифметичні операції з дійсними числами, тому будь-які дійсні числа можуть бути використані як вхідні дані для цієї функції.

Алгоритм вирішення завдання:

1. Приймання вхідних даних A, B, C - значень трьох змінних.
2. Перевірка умови: чи значення сторін (A, B, C) знаходяться в порядку зростання (A < B < C).
3. Якщо умова виконується, то змінні A, B, C помножуються на 2.
4. Якщо умова не виконується, то змінні A, B, C змінюють знак на протилежний (вони стають від'ємними).
5. Виведення нових значень змінних A, B, C.

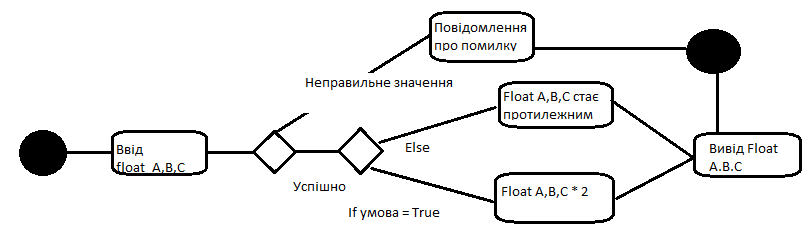


Рисунок 1 – Діаграма діяльності для задачі if16

Лістинг коду вирішення задачі наведено в дод. А (стор. 7-9). Екран роботи програми показаний на рис. Б.1.

Лістинг коду вирішення задачі наведено в дод. А (стор. <номер сторінки>). Екран роботи програми показаний на рис. Б.<номер>.

Завдання 2. Вирішення задачі 10

Функція count\_matrix\_elements:

* Вхідні дані:
  + file\_path: рядок, що містить шлях до файлу з матрицею.
* Вихідні дані:
  + counts: рахунок, що містить кількість входжень кожного символу(точок) ('0', '1', '#') у ту чи іншу зону матриці, згідно їх координат.

Функція task\_2:

* Вхідні дані: відсутні (введення шляху до файлу з матрицею здійснюється користувачем під час виконання програми).
* Вихідні дані: виведення кількості входжень кожного символу(точки) ('0', '1', '#') у матрицю за допомогою функції count\_matrix\_elements.

Опис змінних, які використовуються у цьому коді:

* file\_path: рядок, що містить шлях до файлу з матрицею.
* matrix: матриця, яка представляє собою геометричні фігури на площині х,у з файлу txt.
* counts: містить кількість входжень кожного символу ('0', '1', '#') у матрицю.

Типи даних та діапазони допустимих значень:

* file\_path: рядок - шлях до файлу.
* matrix: список списків рядків - представлення матриці з файлу.
* counts: рахунок, де ключі - це символи ('0', '1', '#'), а значення - це цілі числа, що відображають кількість входжень кожного символу у матрицю.

Алгоритм вирішення

1. Запит від користувача шляху до файлу з матрицею.
2. Виклик функції count\_matrix\_elements з отриманим шляхом до файлу.
3. Вивід результатів підрахунку кількості входжень кожного символу ('0', '1', '#') у зони геометричних фігур на матриці.

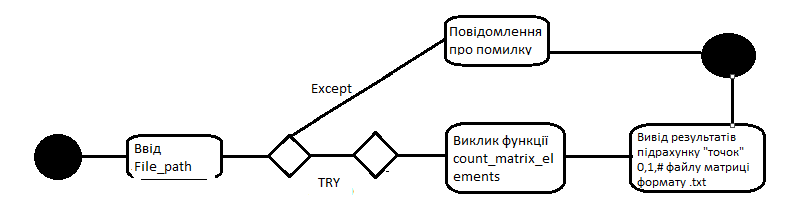


Рисунок 2 – Діаграма діяльності для задачі 10

Лістинг коду вирішення задачі наведено в дод. А (стор. 7-9). Екран роботи програми показаний на рис. Б.2 та 2.1.

Завдання 3. Вирішення задачі 8

Вхідні дані:

* n: int - ціле число, лічильник циклу
* s: float - дійсне число, сума ряду
* u: float - дійсне число, поточний член ряду
* e: float - дійсне число, мале значення для переривання циклу при збіжності
* g: float - дійсне число, велике значення для переривання циклу при розходженні

Вихідні дані:

* s: float - дійсне число, сума збіжного ряду

Алгоритм вирішення

1. Ініціалізувати змінні n, s, u, e, g з відповідними значеннями.
2. Встановити умову циклу, використовуючи while, яка перевірятиме, чи абсолютне значення поточного члена ряду (u) менше за мале значення e і менше за велике значення g.
3. У циклі обчислювати наступний член ряду (u) за допомогою формули, обчислити знаменник ряду та оновити суму ряду (s) шляхом додавання поточного члена.
4. Якщо умова циклу не виконується (або через збіжність, або розходження), вивести суму ряду (s).
5. Обробити виняткові ситуації, такі як ділення на нуль або інші помилки, і вивести відповідне повідомлення.

Зокрема, у цьому коді використовується цикл while для обчислення суми збіжного ряду, а також винятковий блок для обробки помилок.

Лістинг коду вирішення задачі наведено в дод. А (стор. 7-9). Екран роботи програми показаний на рис. Б.3.

# ВИСНОВКИ

Під час виконання роботи було вивчено алгоритмічний підхід до обчислення збіжного ряду, закріплено на практиці використання циклів, обробку винятків та рекурсивні функції в Python. Також відпрацьовано навички взаємодії з користувачем через введення шляху до файлу

ДОДАТОК А

Лістинг коду програми до задач <назви та номери задач>

Module.py

<

#TASK1

def task\_1(A, B, C):

"""

Якщо значення сторін в порядку зростання, вони подвояться; В іншому випадку вони стануть протилежними.

Виведіть нові значення A, B, C.

"""

try:

if A < B < C:

A \*= 2

B \*= 2

C \*= 2

else:

A = -A

B = -B

C = -C

print(f"Нові значення: A={A}, B={B}, C={C}")

except Exception as e:

print(f"Помилка: {e}")

#TASK2

def task\_2():

try:

file\_path = input("Введіть шлях до файлу з матрицею: ")

counts = count\_matrix\_elements(file\_path)

print(f"Кількість точок '0', які не потрапляють на жодну фігуру, або область: {counts['0']}")

print(f"Кількість точок '1', що потрапляють на геометричну фігуру: {counts['1']}")

print(f"Кількість точок '#', що потрапляють в зону болотяного кольору 10-го варіанту: {counts['#']}")

except FileNotFoundError:

print("Файл не знайдено. Перевірте правильність шляху до файлу.")

#TASK3

def task\_3():

"""

Дослідіть ряд на збіжність.

"""

try:

n = 1

s = u = 2.0

e = 1e-20 # Мале значення для переривання циклу при збіжності

g = 1e5 # Велике значення для переривання циклу при розходженні

while abs(u) > e and abs(u) < g:

n += 1

if ((n/2)\*\*n) == 0:

break

# Обчислення знаменника ряду

denominator = 1

for i in range(1, n + 1):

denominator \*= (2 \* i - 1)

u = ((-1)\*\*n \* factorial(n)) / (denominator \* 5\*\*n)

s += u

# Обробка виняткових випадків

print(f"Ряд сходиться до: {s}")

except ZeroDivisionError:

print("Ділення на нуль!")

except Exception as e:

print(f"Помилка: {e}")

# Додаткова функція для обчислення факторіала

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

#Функція для підрахунку точок які потрапляють в зону на системі координат

def count\_matrix\_elements(file\_path):

with open(file\_path, 'r') as file:

matrix = [list(line.strip()) for line in file]

counts = {'0': 0, '1': 0, '#': 0}

for row in matrix:

for char in row:

if char in counts:

counts[char] += 1

return counts

>

Script.py

<

import module

#Створення циклу для визову меню після кожного таску

while True:

print("1. Task 1")

print("2. Task 2")

print("3. Task 3")

print("0. Exit")

choice = int(input("Виберіть завдання (0-3): "))

if choice == 0: # Умова виходу з циклу

break

elif choice == 1:

A = float(input("Введіть сторону A: "))

B = float(input("Введіть сторону B: "))

C = float(input("Введіть сторону C: "))

module.task\_1(A, B, C)

elif choice == 2:

module.task\_2()

elif choice == 3:

module.task\_3()

else:

print("Непрвильний вибір. Спробуйте ще раз.")

>

ДОДАТОК Б

Скрін-шоти вікна виконання програми

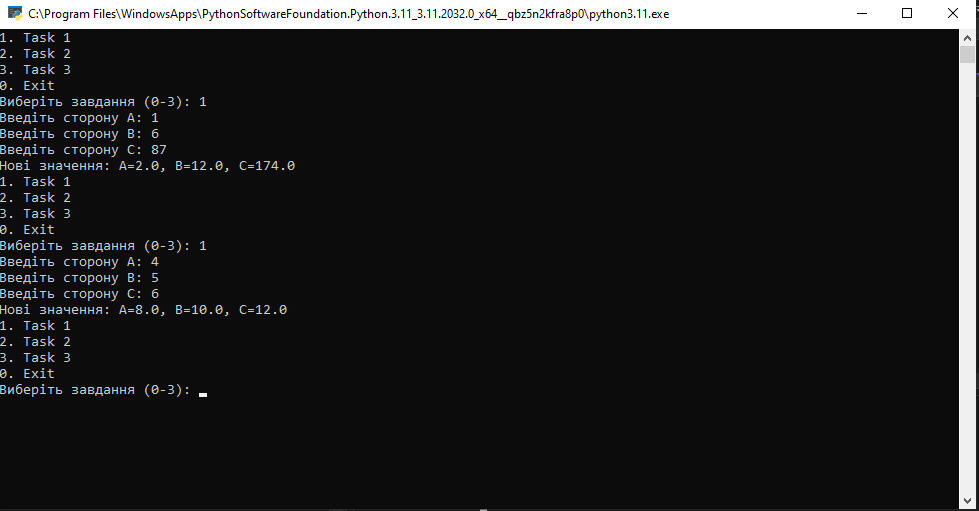


Рисунок Б.1 – Екран виконання програми для вирішення завдання 1

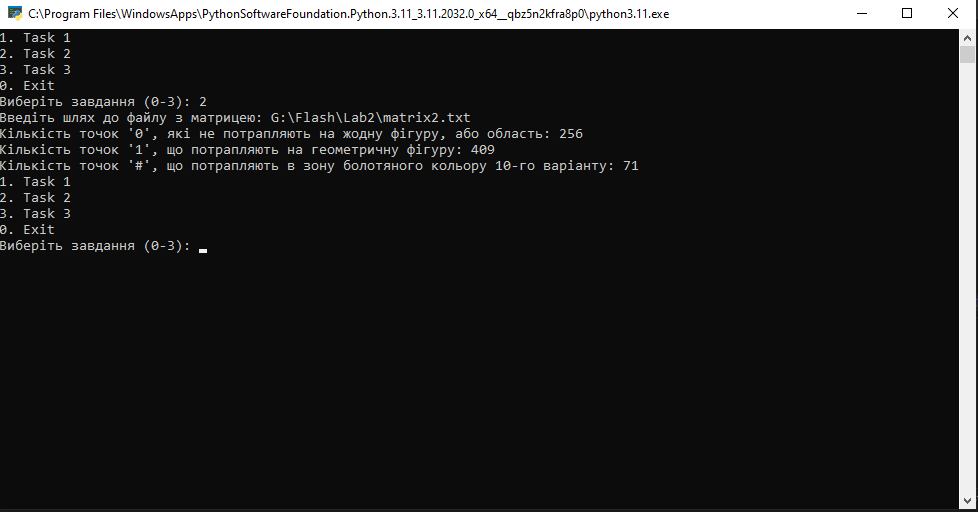
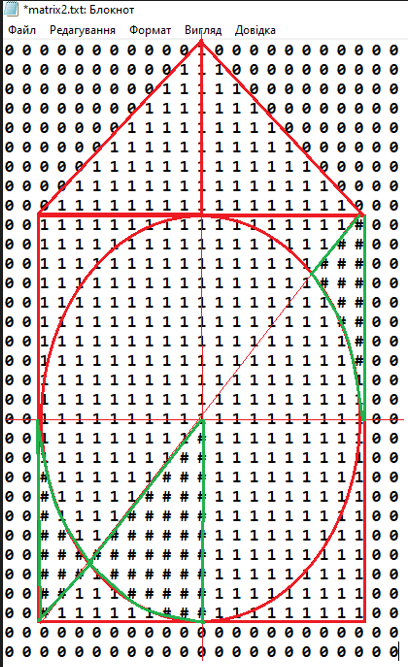


Рисунок Б.2 – Екран виконання програми для вирішення завдання 2

  
Рисунок Б.2.1 – Екран створеної матриці в файлі.txt для вирішення завдання 2

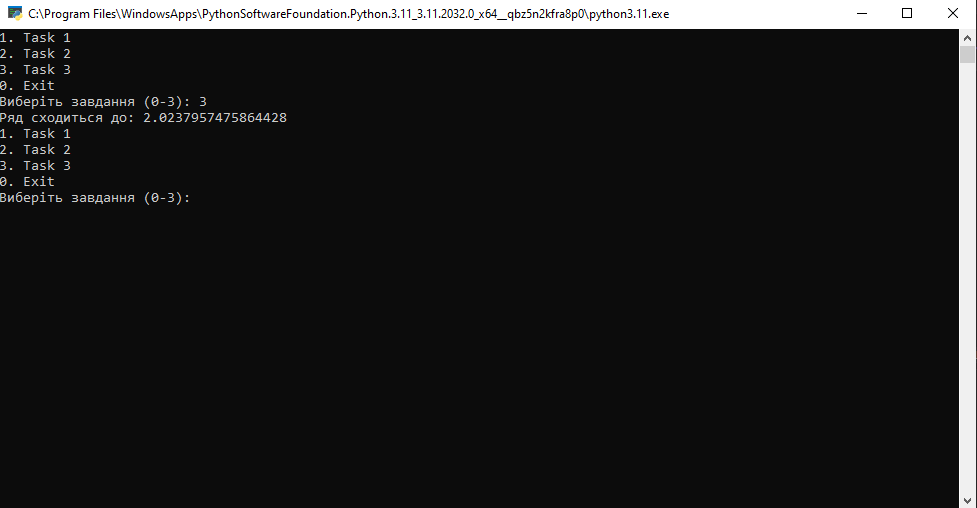


Рисунок Б.3 – Екран виконання програми для вирішення завдання 3