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 . 172.519.СТ.5

Виконав студент гр. _____519СТ_____

_____Пащенко Нікіта___
(підпис, дата) (П.І.Б.)

Перевірив

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

2024

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 на рис.1.

If16.	Дано три змінні дійсного типу: A, B, C. Якщо їх значення впорядковані за зростанням, то подвоїти їх; в іншому випадку замінити значення кожної змінної на протилежне. Вивести нові значення змінних A, B, C.

Рис 1 - Завдання 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). Варіант 15 геометричної області представлений на рис.2.

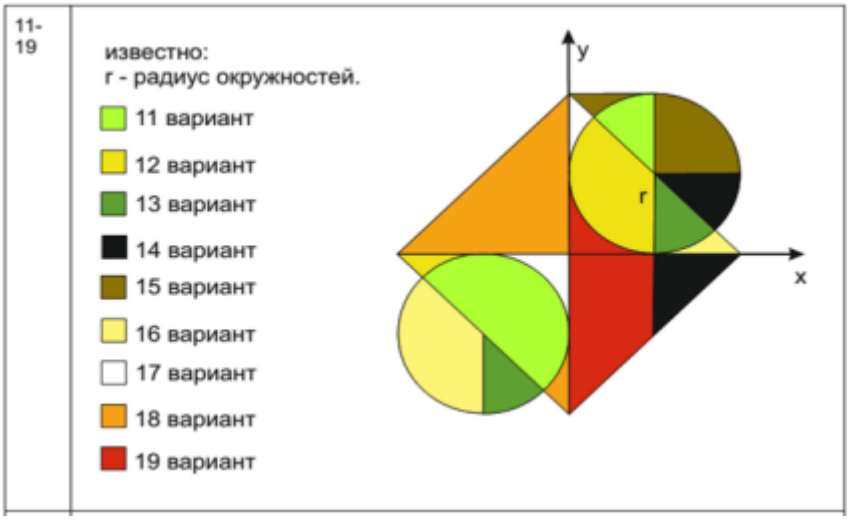

Рис 2 - Завдання 2

Завдання 3. Дослідити ряд на збіжність. Умова закінчення циклу обчислення суми прийняти у вигляді: $|u_n| < E$ або $|u_n| > G$ 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 = 10^{2}... 10^{5}). Варіанти представлено на рис.3.

14	$\sum_{n=1}^{\infty} \frac{x^{2n} + n^{x}}{n!}$

Рис 3 - 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. Перевірка порядку чисел A, B, C
Вхідні дані:

A – перше число, дійсного типу, значення від -10^6 до 10^6.

B – друге число, дійсного типу, значення від -10^6 до 10^6.

C – третє число, дійсного типу, значення від -10^6 до 10^6.

Вихідні дані:
A, B, C – нові значення чисел після перевірки, дійсного типу.

Алгоритм вирішення:

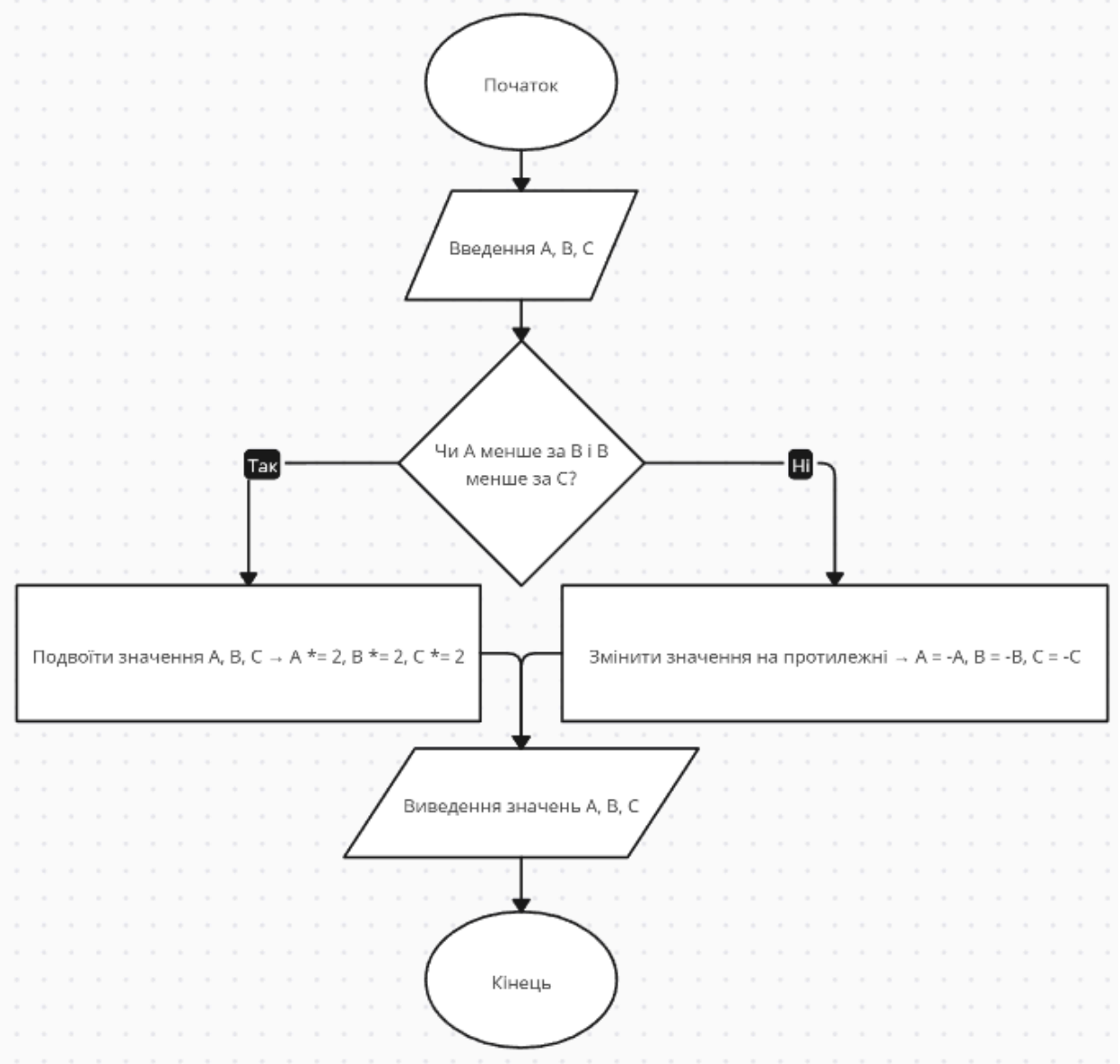

Рисунок 1 - Блок схема до Завдання 1

Завдання 2. Перевірка точок у коричневій області
Вхідні дані:
r – радіус коричневого кола, дійсного типу, значення більше 0.
x, y – координати точок, дійсного типу, значення від $-\ 10^6 6$ до 10^6.
n – кількість точок, цілого типу, значення від 1 до 10^6.
Вихідні дані:
count – кількість точок, що потрапили у коричневу область, цілого типу.

Алгоритм вирішення:

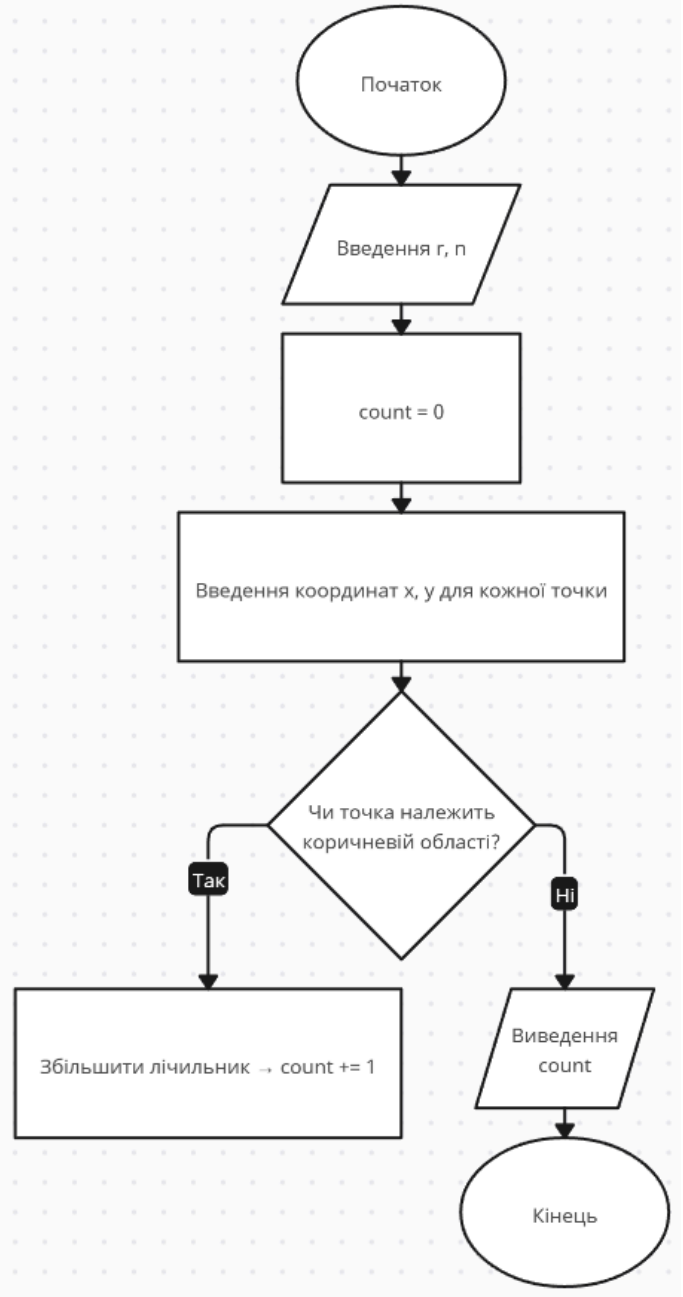

Рисунок 2 - Блок-схема до Завдання 2

Завдання 3. Обчислення суми ряду

Вхідні дані:

x – параметр ряду, дійсного типу, значення від -10^6 до 10^6.

epsilon – точність для перевірки збіжності, дійсного типу, значення від 10^{-20} до 10^{-5}.

G – межа розбіжності, дійсного типу, значення від 10^2 до 10^5.

Вихідні дані:

series_total – сума ряду, дійсного типу.

n – кількість доданків, цілого типу.

Текстове повідомлення про збіжність або розбіжність ряду.

Алгоритм вирішення:

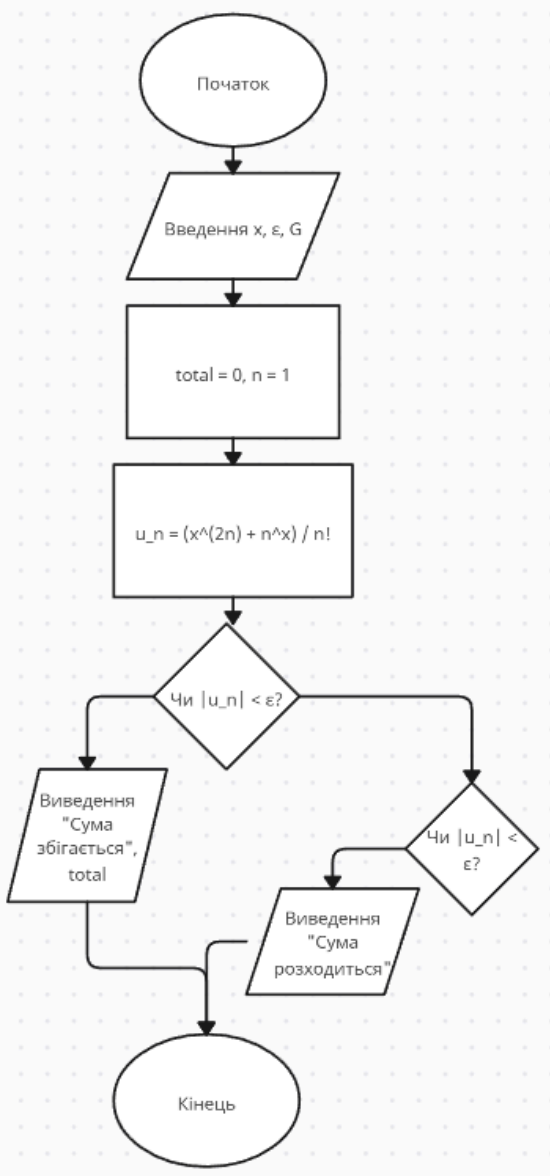

Рисунок 3 - Блок-схема до Завдання 3

Завдання 4. Меню для виконання завдань

Вхідні дані:

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

choice – вибір користувача для виконання певного завдання, цілого типу, значення від 1 до 3.

Змінні для завдань 1, 2 та 3:

A, B, C – параметри для завдання 1, дійсного типу, значення від -10^6 до 10^6.

x – параметр ряду для завдання 3, дійсного типу, значення від -10^6 до 10^6.

epsilon – точність для перевірки збіжності, дійсного типу, значення від 10^{-20} до 10^{-5}.

G – межа розбіжності для завдання 3, дійсного типу, значення від 10^2 до 10^5.

Вихідні дані:

Результати обраного завдання (для завдань 1, 2 або 3):

Для завдання 1: A, B, C – нові значення після обробки, дійсного типу.

Для завдання 2: x, y, count – координати та кількість точок, що потрапляють у область.

Для завдання 3: series_total – сума ряду, дійсного типу; n – кількість доданків, цілого типу.

Текстове повідомлення про завершення роботи або про помилковий вибір.

Алгоритм вирішення:

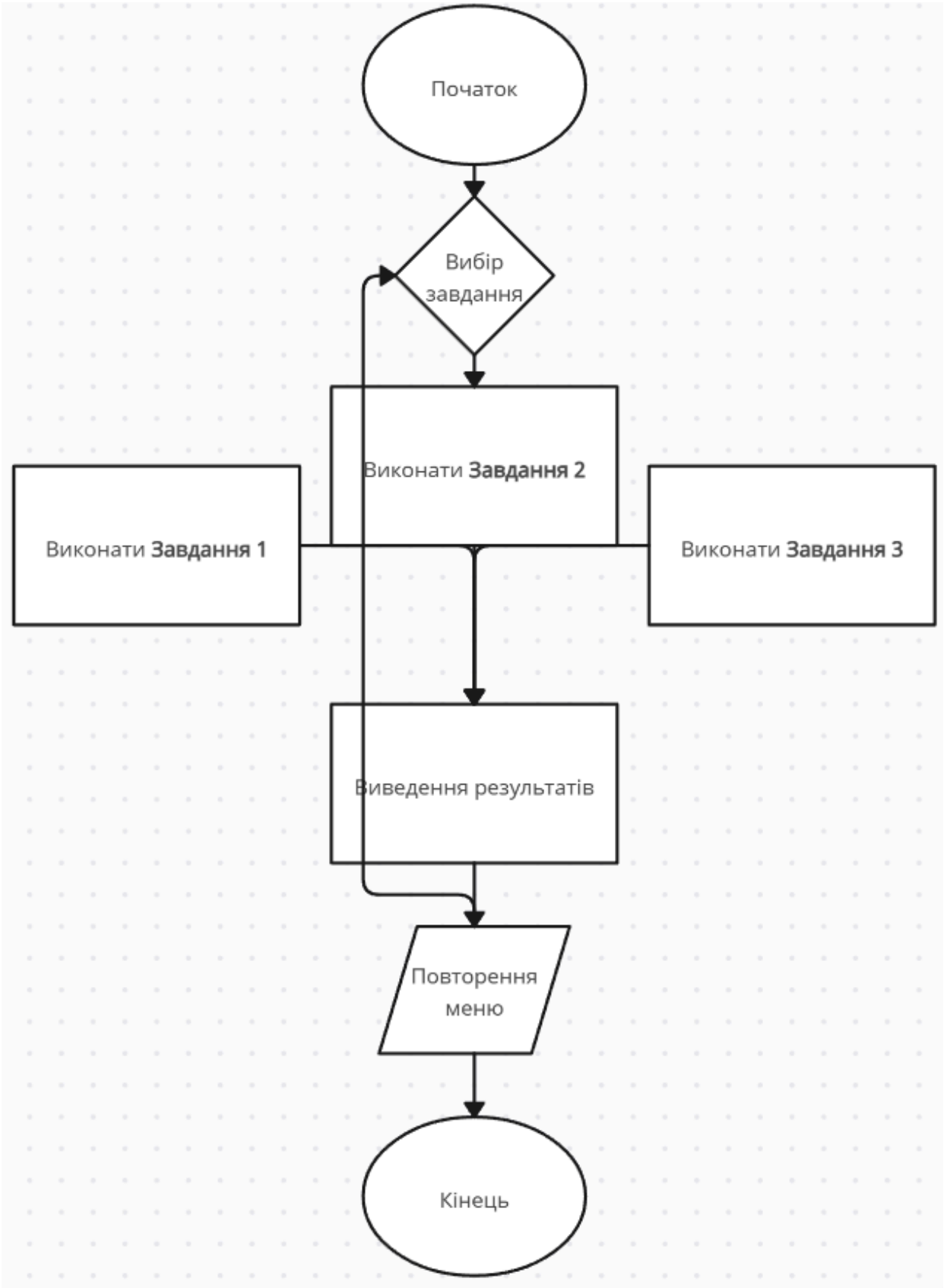

Рисунок 4 - Блок-схема до Завдання 4

Лістинг коду вирішення задачі наведено в дод. А (стор. 10). Екран роботи програми показаний на рис. Б. (стор. 12).

ВИСНОВКИ

У цій роботі було розроблено та реалізовано алгоритми для чотирьох завдань: обробка чисел із розгалуженням, перевірка координат точок, обчислення суми ряду та організація циклічного меню. Кожне завдання структуроване у вигляді блок-схем і програмного коду, що забезпечує зручність виконання та наочність розв'язання поставлених задач.

ДОДАТОК А

Лістинг коду програми до задачі

```
<
   import math

# Завдання 1: Перевірка порядку чисел A, B, C
def process_variables():
    A = float(input("Введіть значення A: "))
    B = float(input("Введіть значення B: "))
    C = float(input("Введіть значення C: "))
    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}")

# Завдання 2: Перевірка точок у коричневій області
def count_points_in_brown_area():
    r = float(input("Введіть радіус кола r: "))
    n_points = int(input("Введіть кількість точок: "))
    points = []
    count = 0
    for i in range(n_points):
        x = float(input(f"Введіть x{i+1}: "))
        y = float(input(f"Введіть y{i+1}: "))
        if x >= 0 and y >= 0 and x**2 + y**2 <= r**2 and y >= x:
            count += 1
    print(f"Кількість точок у коричневій області: {count}")

# Завдання 3: Обчислення суми ряду
def series_sum():
    x = float(input("Введіть значення x: "))
    epsilon = float(input("Введіть ε (точність, наприклад 1e-5): "))
    G = float(input("Введіть G (межа для розбіжності, наприклад 1e3): "))
    n = 1
    series_total = 0.0
    while True:
        u_n = (x**(2*n) + n**x) / math.factorial(n)
        if abs(u_n) < epsilon:
            print("Ряд сходиться.")
            break
        if abs(u_n) > G:
            print("Ряд розходиться.")
            break
        series_total += u_n
        n += 1
```

```
    print(f"Сума ряду: {series_total:.10f}")
    print(f"Кількість доданків: {n}")

# Головне меню
while True:
    print("\n*** ГОЛОВНЕ МЕНЮ ***")
    print("1. Завдання 1: Перевірка порядку чисел")
    print("2. Завдання 2: Перевірка точок у коричневій області")
    print("3. Завдання 3: Обчислення суми ряду")
    print("4. Вихід")

    choice = input("Виберіть завдання (1-4): ")

    if choice == '1':
        print("\nВИКОНАННЯ ЗАВДАННЯ 1")
        process_variables()
    elif choice == '2':
        print("\nВИКОНАННЯ ЗАВДАННЯ 2")
        count_points_in_brown_area()
    elif choice == '3':
        print("\nВИКОНАННЯ ЗАВДАННЯ 3")
        series_sum()
    elif choice == '4':
        print("Програма завершена. До побачення!")
        break
    else:
        print("Неправильний вибір. Спробуйте ще раз.")

>
```

ДОДАТОК Б

Скрін-шоти вікна виконання програми

```
ВИКОНАННЯ ЗАВДАННЯ 1
Введіть значення A: 3
Введіть значення B: 4
Введіть значення C: 5
Нові значення: A = 6.0, B = 8.0, C = 10.0
```

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

```
ВИКОНАННЯ ЗАВДАННЯ 2
Введіть радіус кола r: 5
Введіть кількість точок: 2
Введіть x1: 2
Введіть y1: 3
Введіть x2: 0
Введіть y2: 0
Кількість точок у коричневій області: 2
```

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

```
ВИКОНАННЯ ЗАВДАННЯ 3
Введіть значення x: 2
Введіть ε (точність, наприклад 1e-5): 1e-6
Введіть G (межа для розбіжності, наприклад 1e3): 1e3
Ряд сходиться.
Сума ряду: 59.0347131331
Кількість доданків: 20
```

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

```
*** ГОЛОВНЕ МЕНЮ ***
1. Завдання 1: Перевірка порядку чисел
2. Завдання 2: Перевірка точок у коричневій області
3. Завдання 3: Обчислення суми ряду
4. Вихід
Виберіть завдання (1-4):
```

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