МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

Дисципліна

«Якість програмного забезпечення та тестування»

Лабораторна робота №9

на тему

«Тестування методом «Чорного ящика». Способи діаграм причин – наслідків»

Виконав:	Мішак Максим		**Перевірила**:	Карнаух Тетяна Олександрівна
Група	ІПЗ-33		Дата перевірки	
Форма навчання	денна		Оцінка	
Спеціальність	121			
2023				

Завдання :

Згідно списку академічної групи обрати завдання з таблиці. Перші два стовпчики містить фігури на площині, третій – характеристики. Завдання полягає в наступному – проаналізувати масиви координат з додатку до роботи, щодо положення точок відносно базових фігур. Крім цього треба створити програмний код, що здійснить перевірку положення точки щодо фігури. Варто враховувати, що запропоновані точки не враховують граничних умов – їх необхідно задавати самостійно. За даними розрахунків будується діаграма Ісікави.

Хід роботи :

Для того , щоб вирішити проблему диференціації точок , тобто віднести їх до однієї з трьох груп :

1. Всередині фігури
2. За межами фігури
3. На межі фігури

було побудовано застосунок , який за допомогою алгоритму розділяє координати точок , які надані в таблиці . Код застосунку виглядає так :

```
def is_point_inside_square(point, center, side_length):
  half_side = side_length / 2
  x_min, x_max = center[0] - half_side, center[0] + half_side
  y_min, y_max = center[1] - half_side, center[1] + half_side
  return (x_min <= point[0] <= x_max) and (y_min <= point[1] <= y_max)
table_1_raw = """
2,3 0,1 2,4 0,0 1,2 0,2 2,2 1,1
0,9 2,0 3,3 2,3 1,3 2,4 3,2 1,5
1,0 3,1 1,1 2,0 2,1 2,8 1,0 4,3
0,2 1,5 1,3 1,0 2,2 1,2 1,1 6,6
2,2 5,2 1,3 0,2 3,1 2,7 1,2 3,3
1,1 1,4 3,1 1,0 2,2 0,9 2,1 2,5
0,0 4,4 2,3 5,0 1,3 2,3 2,0 4,1
5,2 3,0 0,2 0,5 1,2 2,3 3,2 0,3
3,6 2,6 3,1 3,3 1,1 2,3 4,2 0,2
2,0 6,0 5,1 2,2 1,2 0,3 3,1 0,1
0,4 0,1 6,4 4,2 2,3 1,0 1,5 1,0
2,2 4,0 2,4 2,1 1,2 1,0 2,3 2,0
3,1 3,3 3,2 1,3 2,2 2,3 4,0 3,0
1,3 2,2 1,2 0,1 3,1 2,3 0,2 4,2
```

```
0,2 1,1 5,5 2,6 2,1 0,0 6,2 3,1
2,0 2,2 4,3 4,2 0,2 2,1 3,3 2,4
"""
table_2_raw = """
2,0 6,0 5,1 2,2 1,2 0,3 3,1 0,1
0,2 1,1 5,5 2,6 2,1 0,0 6,2 3,1
2,2 5,2 1,3 0,2 3,1 2,7 1,2 3,3
0,0 4,1 2,3,1 3,3 3,2 1,3 2,1 2,2 4,0 3,0
2,3 0,1 2,4 0,0 1,2 0,2 2,2 1,1
1,1 1,4 3,1 1,0 2,2 0,9 2,1 2,5
0,4 0,1 6,4 4,2 2,3 1,0 1,5 2,0
2,0 2,2 4,3 4,2 0,2 2,1 3,3 2,4
5,2 3,0 0,2 0,5 1,2 2,3 3,2 0,3
0,2 1,5 1,2 1,0 2,2 1,2 1,1 6,6
3,6 2,6 3,1 2,3 1,1 3,3 4,2 0,2
2,2 4,0 2,0 2,1 1,2 1,0 2,3 2,0
1,3 2,2 1,2 0,1 3,1 2,3 0,2 4,2
1,0 3,1 1,1 2,0 2,1 2,8 1,0 4,3
0,9 2,0 3,3 2,3 1,3 2,4 3,2 1,5
"""
table_1 = [tuple(map(int, coords.split(','))) for coords in table_1_raw.split()]
table_2 = [tuple(map(int, coords.split(','))) for coords in table_2_raw.split()]
larger_circle_center = (0, 0)
larger_circle_side = 12
smaller_circle_center = (0, 0)
smaller_circle_side = 6
points_table_1 = {'inside_larger': [], 'inside_smaller': [], 'outside': [], 'boundary':
[]}
points_table_2 = {'inside_larger': [], 'inside_smaller': [], 'outside': [], 'boundary':
[]}

for point in table_1:
  if is_point_inside_square(point, larger_circle_center, larger_circle_side):
    if is_point_inside_square(point, smaller_circle_center, smaller_circle_side):
      points_table_1['inside_smaller'].append(point)
    else:
      points_table_1['inside_larger'].append(point)
```

```
    else:
        points_table_1['outside'].append(point)

for point in table_2:
    if is_point_inside_square(point, larger_circle_center, larger_circle_side):
        if is_point_inside_square(point, smaller_circle_center, smaller_circle_side):
            points_table_2['inside_smaller'].append(point)
        else:
            points_table_2['inside_larger'].append(point)
    else:
        points_table_2['outside'].append(point)

print(points_table_1)
print(points_table_2)
```

Результати виконання програми :

{'inside_larger': [(2, 4), (2, 4), (1, 5), (4, 3), (1, 5), (6, 6), (5, 2), (1, 4), (2, 5), (4, 4), (5, 0), (4, 1), (5, 2), (0, 5), (3, 6), (2, 6), (4, 2), (6, 0), (5, 1), (0, 4), (6, 4), (4, 2), (1, 5), (4, 0), (2, 4), (4, 0), (4, 2), (5, 5), (2, 6), (6, 2), (4, 3), (4, 2), (2, 4)], 'inside_smaller': [(2, 3), (0, 1), (0, 0), (1, 2), (0, 2), (2, 2), (1, 1), (2, 0), (3, 3), (2, 3), (1, 3), (3, 2), (1, 0), (3, 1), (1, 1), (2, 0), (2, 1), (1, 0), (0, 2), (1, 3), (1, 0), (2, 2), (1, 2), (1, 1), (2, 2), (1, 3), (0, 2), (3, 1), (1, 2), (3, 3), (1, 1), (3, 1), (1, 0), (2, 2), (2, 1), (0, 0), (2, 3), (1, 3), (2, 3), (2, 0), (3, 0), (0, 2), (1, 2), (2, 3), (3, 2), (0, 3), (3, 1), (3, 3), (1, 1), (2, 3), (0, 2), (2, 0), (2, 2), (1, 2), (0, 3), (3, 1), (0, 1), (0, 1), (2, 3), (1, 0), (1, 0), (2, 2), (2, 1), (1, 2), (1, 0), (2, 3), (2, 0), (3, 1), (3, 3), (3, 2), (1, 3), (2, 2), (2, 3), (3, 0), (1, 3), (2, 2), (1, 2), (0, 1), (3, 1), (2, 3), (0, 2), (0, 2), (1, 1), (2, 1), (0, 0), (3, 1), (2, 0), (2, 2), (0, 2), (2, 1), (3, 3)], 'outside': [(0, 9), (2, 8), (2, 7), (0, 9)], 'boundary': []}

{'inside_larger': [(6, 0), (5, 1), (5, 5), (2, 6), (6, 2), (5, 2), (4, 1), (4, 0), (2, 4), (1, 4), (2, 5), (0, 4), (6, 4), (4, 2), (1, 5), (4, 3), (4, 2), (2, 4), (5, 2), (0, 5), (1, 5), (6, 6), (3, 6), (2, 6), (4, 2), (4, 0), (4, 2), (4, 3), (2, 4), (1, 5)], 'inside_smaller': [(2, 0), (2, 2), (1, 2), (0, 3), (3, 1), (0, 1), (0, 2), (1, 1), (2, 1), (0, 0), (3, 1), (2, 2), (1, 3), (0, 2), (3, 1), (1, 2), (3, 3), (0, 0), (2, 3, 1), (3, 3), (3, 2), (1, 3), (2, 1), (2, 2), (3, 0), (2, 3), (0, 1), (0, 0), (1, 2), (0, 2), (2, 2), (1, 1), (1, 1), (3, 1), (1, 0), (2, 2), (2, 1), (0, 1), (2, 3), (1, 0), (2, 0), (2, 0), (2, 2), (0, 2), (2, 1), (3, 3), (3, 0), (0, 2), (1, 2), (2, 3), (3, 2), (0, 3), (0, 2), (1, 2), (1, 0), (2, 2), (1, 2), (1, 1), (3, 1), (2, 3), (1, 1), (3, 3), (0, 2), (2, 2), (2, 0), (2, 1), (1, 2), (1, 0), (2, 3), (2, 0), (1, 3), (2, 2),

(1, 2), (0, 1), (3, 1), (2, 3), (0, 2), (1, 0), (3, 1), (1, 1), (2, 0), (2, 1), (1, 0), (2, 0), (3, 3), (2, 3), (1, 3), (3, 2)], 'outside': [(2, 7), (0, 9), (2, 8), (0, 9)], 'boundary': []}

Аналіз виконання :

З результатів виконання можемо побачити , що программа коректно розділяє точки , та в результаті виводить списки : 'inside_larger' - який відповідає першій групі , 'outside' - який відповідає другій групі , та 'boundary' , який відповідає третій групі .

Побудова графа причинно-наслідкових зв’язків :

_____

–Проблема–

_____

/

/1.Розмір квадрата

/——1.1 Довжина сторони більшого квадрату

/——1.2 Довжина меншого квадрату

/

/2.Центр квадрата

/——2.1 Центр більшого квадрату

/——2.2 Центр меншого квадрату

/

/3.Діаметр кола

/——3.1 Центр більшого кола

/——3.2 Центр меншого кола

/

/4.Координати точок

/——4.1 Координати точок з таблиці 1

/——4.2 Координати точок з таблиці 2

/

/5.Умови

/——5.1 Точка в межах фігури

/——5.2 Точка на межі фігури

/——5.3 Точка за межами фігури

Побудувавши причинно-наслідковий графік на основі цього контуру, ви зможете візуалізувати фактори, які впливають на положення точок відносно базових фігур. Це може допомогти вам краще зрозуміти проблему і визначити області для поліпшення або подальшого аналізу. Генерація таблиці рішень. Причини розглядаються як умови, а наслідки –

як дії. Ось інтерпретація таблиці рішень з номерами стовпців, умовами, вторинними причинами та діями:

Column #	1	2	3	4	5	6
Cause	1	2	3	4	5	6
C1	1	1	1	0	0	0
C2	0	0	0	1	1	1
C3	x	x	x	1	0	0
C4	x	x	x	0	1	0
C5	x	x	x	0	0	1
C6	1	0	0	x	x	x
C7	0	1	0	x	x	x
C8	0	0	1	x	x	x
S. Cause	11	12	13	14	15	16
Action	101	102	103	104	105	106

Перетворення кожного стовпця таблиці в тестовий варіант. У нашому прикладі 4 тестових варіанту.

1. Тестовий приклад 1:
    a. Причина 1: Істина
    b. Причина 2: False
    c. Причина 3: x (будь-яке значення)
    d. Вторинна причина 11: True
    e. Дія 101: True
    f. Дія 102: Неправда
    g. Дія 103: False
    h. Дія 104: Неправда 2.

Тестовий приклад 2:

a. Причина 1: Істина
b. Причина 2: False
c. Причина 3: x (будь-яке значення)
d. Вторинна причина 12: True
e. Дія 101: Неправда
f. Дія 102: True
g. Дія 103: False
h. Дія 104: Неправда 3.

Тестовий приклад 3:

a. Причина 1: True
b. Причина 2: False
c. Причина 3: x (будь-яке значення)
d. Вторинна причина 13: True
e. Дія 101: Неправда
f. Дія 102: False
g. Дія 103: Істина
h. Дія 104: False

4. Тестовий приклад 4:

a. Причина 1: False
b. Причина 2: Істина
c. Причина 3: Істина
d. Причина 4: Хибна
e. Вторинна причина 14: Істина
f. Дія 101: Хибна
g. Дія 102: Хибна
h. Дія 103: Неправда
i. Дія 104: Істина

Тестування комбінаці умов :

TB1

Points in Table 1:

Inside larger: [(0, 1), (2, 4), (0, 0), (0, 2)]
Inside smaller: [(2, 3), (1, 2), (2, 2), (1, 1)]
Outside: []
Boundary: []

Points in Table 2:

Inside larger: [(2, 0), (0, 3), (0, 1)]
Inside smaller: [(2, 2), (1, 2), (3, 1)]

Outside: [(6, 0), (5, 1)]
Boundary: []

TB2

Points in Table 1:
Inside larger: [(2, 0), (2, 4)]
Inside smaller: [(3, 3), (2, 3), (1, 3), (3, 2)]
Outside: [(0, 9), (1, 5)]
Boundary: []

Points in Table 2:
Inside larger: [(0, 2), (0, 0)]
Inside smaller: [(1, 1), (2, 1), (3, 1)]
Outside: [(5, 5), (2, 6), (6, 2)]
Boundary: []

TB3

Points in Table 1:
Inside larger: [(1, 0), (2, 0), (1, 0), (4, 3)]
Inside smaller: [(3, 1), (1, 1), (2, 1)]
Outside: [(2, 8)]
Boundary: []

Points in Table 2:
Inside larger: [(0, 2)]
Inside smaller: [(2, 2), (1, 3), (3, 1), (1, 2), (3, 3)]
Outside: [(5, 2), (2, 7)]
Boundary: []

TB4

Points in Table 1:
Inside larger: [(0, 2), (1, 0)]
Inside smaller: [(1, 3), (2, 2), (1, 2), (1, 1)]
Outside: [(1, 5), (6, 6)]
Boundary: []

Points in Table 2:
Inside larger: [(0, 1), (2, 4), (0, 0), (0, 2)]
Inside smaller: [(2, 3), (1, 2), (2, 2), (1, 1)]
Outside: []
Boundary: []

Висновок :

На лабораторній роботі було виконано завдання по розробці та тестуванню додатку , який розділяє точки на категорії . Точки розташовані на декартовій площині . Для розділення алгоритм порівнює фактичні координати точки з допустимими координатами , які належать до фігури , яка задана в програмі . За допомогою двох циклів порівняння відбувається з кожною точкою з двох таблиць . Було побудовано діаграму Ісікави та розписано її значення в текстовому форматі . Також наведені тест-кейси для 4х точок з датасету