# ЛАБОРАТОРНАЯ РАБОТА №1

# ИССЛЕДОВАНИЕ СПОСОБОВ АНАЛИЗА ОБЛАСТЕЙ ЭКВИВАЛЕНТНОСТИ И ПОСТРОЕНИЯ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Работы выполнил: Мовенко Константин, ИС/б-21-2-о, Вариант - 8

Цель работы

Исследовать способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Приобрести практические навыки составления и построения тестовых последовательностей.

Задачи

По варианту задаются требования к программам. Для каждой из них необходимо:

1. Написать программу, выполняющую заданные действия;
2. Определить области эквивалентности входных данных;
3. Составить примеры тестовых последовательностей;

Вариант задания

Задача 1. Дана квадратная матрица. Выполнить поворот этой матрицы на 90\*k градусов, где k − целое число.

Задача 2. Дана строка. Выяснить, верно ли, что в строке имеются пять идущих подряд букв 'е'.

Задача 3. Программа, которая подсчитывает количество пустых строк в текстовом файле.

Текст программы

Листинг 1 – main.py

import task1

import task2

import task3

*# Выполнение первого задания*

task1.execute(-5)

task2.execute()

task3.execute('input.txt')

Листинг 2 – task1.py

*# Ручной ввод матрицы (построчно)*

def enter\_matrix():

matrix = []

print('Введите матрицу (конец - end):')

s = input()

while s != 'end':

matrix.append(s.split())

s = input()

return matrix

*# Вывод матрицы*

def print\_matrix(matrix):

for i in matrix:

print(\*i)

*# Поворот матрицы на k градусов*

def turn\_matrix(matrix, k):

if k >= 0:

rotations = k % 4

else:

rotations = 4 - (-k % 4)

print(rotations)

for i in range(rotations):

matrix = list(map(lambda x: list(x), zip(\*matrix[::-1])))

return matrix

def execute(k):

print('---ЗАДАНИЕ 1---')

m = enter\_matrix()

m = turn\_matrix(m, k)

print('\nМатрица после k=' + str(k) + ' поворотов')

print\_matrix(m)

print()

Листинг 3 – task2.py

def execute():

print('---ЗАДАНИЕ 2---')

print('Введите строку:')

s = input()

if 'e' \* 5 in s:

print('Подстрока "eeeee" найдена')

else:

print('Подстрока не найдена')

print()

Листинг 4 – task3.py

def execute(filename):

print('---ЗАДАНИЕ 3---')

file = open(filename)

count = 0

for line in file:

if line.strip() == '':

count += 1

print('Найдено пустых строк:', str(count))

file.close()

print()

Ход работы

Области эквивалентности для задания 1:

1. По размеру матрицы:
   1. Матрица из одного элемента;
   2. Матрица из множества элементов;
2. По значению k:
   1. ;
   2. ;
   3. ;

Относительно них были составлены тестовые последовательности задания 1 (Таблица 1).

|  |  |  |
| --- | --- | --- |
| Соответствие классам | Тестовая последовательность | Ожидаемый результат |
| 1a |  |  |
| 1b, 2a |  |  |
| 1b, 2b |  |  |
| 1b, 2c |  |  |

Таблица 1 – Тестовые примеры для задания 1

Каждая тестовая последовательность была протестирована через программу. В результате все полученные результаты совпали с ожидаемыми.

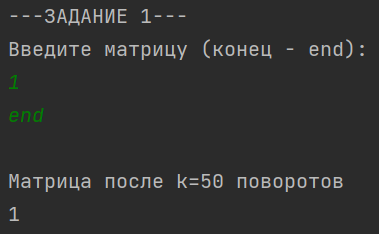


Рисунок 1 – Тестирование класса 1a, задание 1

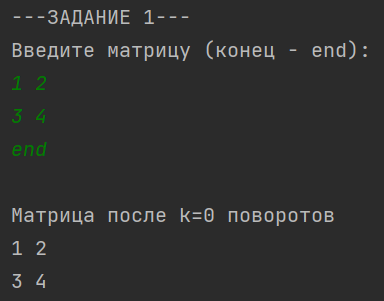


Рисунок 2 – Тестирование классов 1b 2a, задание 1

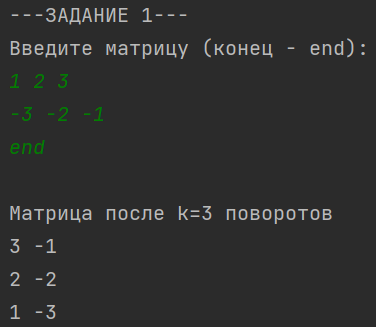


Рисунок 3 – Тестирование классов 1b 2b, задание 1

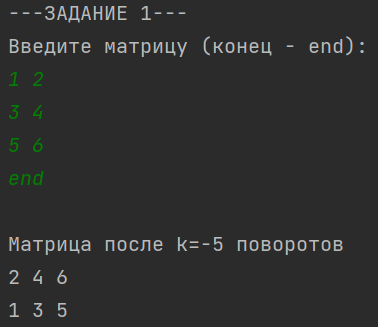


Рисунок 4 – Тестирование классов 1b 2c, задание 1

Были определены области эквивалентности для задания 2:

1. По содержанию строки:
   1. Строка пуста;
   2. Строка состоит из одного и более элементов;
2. По наличию в строке ряда 'eeeee':
   1. Ряд существует, расположен в середине строки;
   2. Ряд существует, расположен в начале строки;
   3. Ряд существует, расположен в конце строки;
   4. Ряд отсутствует в строке;

Относительно них были составлены тестовые последовательности задания 2 (Таблица 2).

|  |  |  |
| --- | --- | --- |
| Соответствие классам | Тестовая последовательность | Ожидаемый результат |
| 1a | «» | false |
| 1b, 2a | «treeeeeesh» | true |
| 1b, 2b | «eeeee rockkk!» | true |
| 1b, 2c | «oh yeeeee» | true |
| 1b, 2d | «HELLO WORD» | false |

Таблица 2 – Тестовые примеры для задания 2

Каждая тестовая последовательность была протестирована через программу. В результате все полученные результаты совпали с ожидаемыми.

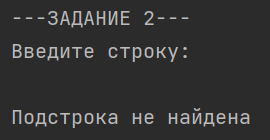


Рисунок 5 – Тестирование класса 1a, задание 2

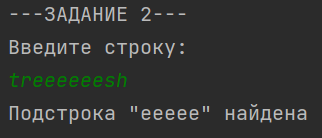


Рисунок 6 – Тестирование классов 1b 2a, задание 2

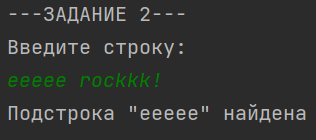


Рисунок 7 – Тестирование классов 1b 2b, задание 2

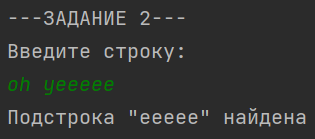


Рисунок 8 – Тестирование классов 1b 2c, задание 2

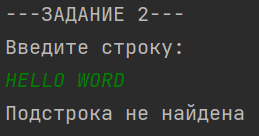


Рисунок 9 – Тестирование классов 1b 2d, задание 2

Были определены области эквивалентности для задания 3:

1. По содержанию файла:
   1. Файл пустой;
   2. В файле одна строка;
   3. В файле множество строк;
2. По количеству пустых строк:
   1. Пустых строк нет;
   2. Есть одна и более пустых строк;
3. По доступу к файлу:
   1. Файл существует и доступен программе;
   2. Файл недоступен или не существует;

Относительно них были составлены тестовые последовательности задания 3 (Таблица 3).

|  |  |  |
| --- | --- | --- |
| Соответствие классам | Тестовая последовательность | Ожидаемый результат |
| 1a, 2a, 3a | «» (input\_1.txt) | 0 |
| 1b, 2a, 3a | «no lines there» (input\_2.txt) | 0 |
| 1b, 2b, 3a | « » (input\_3.txt) | 1 |
| 1c, 2a, 3a | «Day day de mm hm  Da da da ba ba  Okay»  (input\_4.txt) | 0 |
| 1c, 2b, 3a | «line  Line  » (input\_5.txt) | 2 |
| 3b | Файл не существует (input\_6.txt) | Сообщение об ошибке |

Таблица 3 – Тестовые примеры для задания 3

Каждая тестовая последовательность была протестирована через программу. В результате все полученные результаты совпали с ожидаемыми.

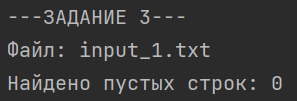


Рисунок 10 – Тестирование класса 1a (2a, 3a), задание 3

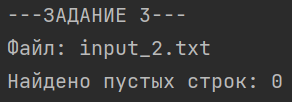


Рисунок 11 – Тестирование классов 1b 2a 3a, задание 3

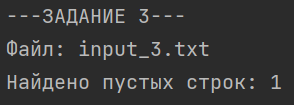


Рисунок 12 – Тестирование класса 1b 2b 3a, задание 3

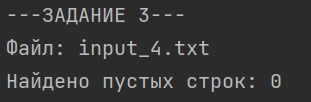


Рисунок 13 – Тестирование класса 1c 2a 3a, задание 3

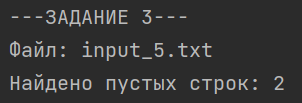


Рисунок 14 – Тестирование класса 1c 2b 3a, задание 3

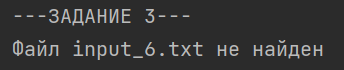


Рисунок 15 – Тестирование класса 3b, задание 3

Вывод

В ходе работы были приобретены практические навыки тестирования программ через классификацию областей эквивалентности их входных данных. При написании программ были определены области эквивалентности для различных типов задач, для областей были составлены тестовые последовательности, на основе которых проводилось тестирование.

Полученные знания и навыки позволяют наиболее ёмко определять основные вариации входных данных, находить особые случаи и эффективно проводить тестирование программ.