Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЕТ

по лабораторной работе №2

ИССЛЕДОВАНИЕ СПОСОБОВ СТРУКТУРНОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Собченко М.В.

Севастополь

2023

# ЦЕЛЬ РАБОТЫ

Исследовать основные подходы к структурному тестированию программного обеспечения. Приобрести практические навыки построения графа потоков управления и определения независимых ветвей программы.

# ЗАДАНИЕ

Для варианта задания из л.р. №1 (Вариант − 8):

* 1. Написать программу, выполняющую заданные действия;
  2. Построить граф потоков управления;
  3. Вычислить цикломатическое число для построенного графа потоков управления;
  4. Определить независимые ветви программы;

Задачи для программ:

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.

# ХОД РАБОТЫ

Был построен граф потоков управления для функции turn\_matrix, выполняющей задание №1 (Рисунок 1).

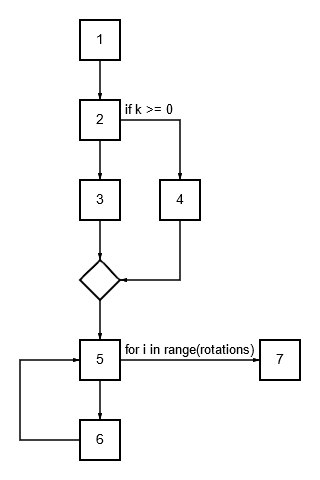


Рисунок 1 – Граф потоков управления функции для задания 1

Граф состоит из 9 дуг и 7 узлов. По этим значениям для него было вычислено цикломатическое число:

Граф имеет 4 независимых ветви:

1. 1, 2, 3, 5, 7;
2. 1, 2, 4, 5, 7;
3. 1, 2, 3, 5, 6, 5, 7;
4. 1, 2, 4, 5, 6, 5, 7;

Далее был построен граф потоков управления для программы, выполняющей задание №2 (Рисунок 2).

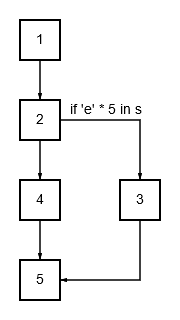


Рисунок 2 – Граф потоков управления программы 2

Граф состоит из 5 узлов и 5 дуг. По этим значениям было вычислено цикломатическое число.

Граф имеет 2 независимых ветви:

1. 1, 2, 3, 5;
2. 1, 2, 4, 5;

Был построен граф потоков управления для программы, выполняющей задание №3 (Рисунок 3).

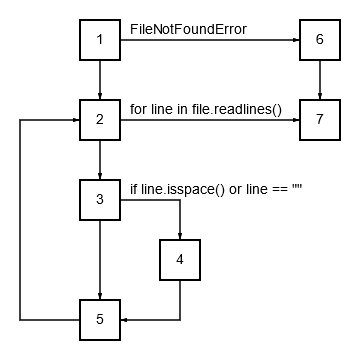


Рисунок 3 – Граф потоков управления программы 3

Граф состоит из 9 дуг и 7 узлов. По данным значениям было вычислено цикломатическое число.

Граф имеет 4 независимых ветви:

1. 1, 2, 7;
2. 1, 2, 3, 5, 2, 7;
3. 1, 2, 3, 4, 5, 2, 7;
4. 1, 6;

# текст программы

Листинг 1 – main.py

import task1  
import task2  
import task3  
  
*# Тестирование различных заданий*n = int(input("Введите номер программы: "))  
match n:  
 case 1: task1.execute(int(input("Введите k: ")))  
 case 2: task2.execute()  
 case 3: task3.execute(input("Введите имя файла: "))

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)  
 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---')  
 try:  
 with open(filename) as file:  
 count = 0  
 for line in file.readlines():  
 if line.isspace() or line == "":  
 count += 1  
 print('Файл: ' + filename)  
 print('Найдено пустых строк:', str(count))  
 except FileNotFoundError:  
 print('Файл ' + filename + ' не найден')  
 print()

# ВЫВОД

В ходе работы был исследован структурный подход к тестированию ПО на примере метода исследования ветвей программы. Были получены практические навыки построения графов потоков управления программ и выделения в них независимых ветвей, на основе которых можно формировать наборы тестовых данных.