Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Рязанский государственный радиотехнический университет

им. В.Ф. Уткина»

Кафедра ЭВМ

Отчет по лабораторной работе №1

Структурное тестирование программного обеспечения

Выполнили:

студенты бригады <8>

Зотов С. С. гр. 3840М.

Алясов А. гр. 340М.

Аксентьев М. гр. 340М.

Проверил:

Доцент Саблина В. А.

Рязань, 2023

Цель работы

Получение знаний об особенностях структурного тестирования программного обеспечения, а также умений и навыков его выполнения для оценки качества программного модуля, реализующего алгоритм поставленной задачи.

Ход работы

Для выполнения лабораторной работы был выбран 18-й вариант в котором необходимо написать и протестировать программу, определяющую является ли введенное число, простым числом Мерсенна. Учитывая, что простое число Мерсенна *M* – это простое число, представимое в виде *𝑀 = 2𝑝 − 1*, где p – другое простое число, необходимо проверить, является ли заданное число x простым числом Мерсенна; результатом является булево значение b – результат проверки. Для решения данной задачи, была разработана следующая программа, позволяющая определить является ли заданное число простым числом Мерсенна. По условиям задачи, Функция принимает число и после проверки возвращает булево значение **True** при положительном результате и **False** при отрицательном.

Рисунок 1 Программа для проверки того, является ли заданное число простым числом Мерсенна

**import** math

# Функция проверки простоты числа

**def** isprime**(**n**):**

i **=** 2

**while** i **<=** math**.**sqrt**(**n**):**

**if** **(**n **%** i**)** **==** 0**:**

**return** **False**

i **=** i **+** 1

**return** **True**

# Фнкция принимает аргумент и приводит его значение

# к целочисленному типу, в случае если привести значение

# к целому типу не возможно, функция завершает свою работу

**def** check\_mersens\_number**(**num**):**

**try:**

number **=** **int(**num**)**

**except** **ValueError:**

**print(**f' "{num}" invalid input. Expected number'**)**

**return** 0

p **=** math**.**log2**(**number **+**1**)**

**if** p **%** 1 **!=** 0**:**

**return** **False**

S **=** 4

i **=** 1

**print(**p**)**

**if** isprime**(int(**p**)):**

**while** i **!=** p**-**1**:**

S **=** **(**S**\*\***2 **-** 2**)** **%** number

i **=**i **+** 1

**if** S **==** 0**:**

**return** **True**

**else:**

**return** **False**

**else:**

**return** **False**

Используя метод структурного тестирования (метод прозрачного ящика) с помощью способа базового пути построили потоковый граф разбив программу на узлы графа.

1. *check\_mersens\_number(num):*
2. *num* Приводится к целому числу?
3. Выход с ошибкой
4. *p = math.log2(number + 1)*
5. *p % 1 !=* *0*
6. Число не является числом Мерсенна
7. *S = 4, i=1*
8. пока i **<=** math**.**sqrt**(**n**)**.
9. **(**n **%** i**)** **==** 0 ?
10. Возвращаем **False**.
11. Пока i **!=** p**-**1.
12. S **=** **(**S**\*\***2 **-** 2**)** **%** number; i **=**i **+** 1.
13. S **==** 0.
14. Возвращаем **True**.
15. Возвращаем **False**.
16. Выход

Используя полученные данные построили потоковый граф(рисунок 2)

С помощью графа нашли следующие независимые пути:

1. 1-2-3-16
2. 1-2-4-5-6-16
3. 1-2-4-5-7-8-9-10-16
4. 1-2-4-5-7-8-9-8-11-12-11-13-14-16
5. 1-2-4-5-7-8-9-8-11-12-11-13-15-16

Просчитали все возможные независимые пути для потокового графа(рисунок 3) определили цикломатическую сложность по следующей формуле V(G) = E – N + 2, где E =21 – количество дуг, N = 16 – количество узлов потокового графа. Из этого следует что цикломатическая сложность равна 7 (21-16+2 = 5+2=7)

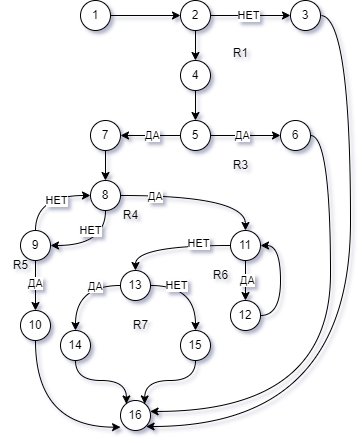


Рисунок 4 Потоковый граф

Для тестирования программы был написан UNIT тест (рисунок 3)

Рисунок 5 UNIT тест

**class** **TestCheckMersensNumber(**unittest**.**TestCase**):**

**def** setUp**(**self**):**

self**.**chmn **=** check\_mersens\_number

**def** test\_MersensPrimeNum**(**self**):**

self**.**assertEqual**(**self**.**chmn**(**8191**),** **True)**

**def** test\_MersensNum**(**self**):**

self**.**assertEqual**(**self**.**chmn**(**4095**),** **False)**

**def** test\_NotMersensNum**(**self**):**

self**.**assertEqual**(**self**.**chmn**(**8056**),** **False)**

**def** test\_NotMersensPrimeNum**(**self**):**

self**.**assertEqual**(**self**.**chmn**(**17**),** **False)**

**def** test\_MersensNumWithNotNumber**(**self**):**

self**.**assertEqual**(**self**.**chmn**(**"twuenty"**),** **False)**

В данном тесте проверили все 5-ть независимых путей, все тесты прошли успешно

[Running] python -u "c:\Users\zotov\Workplaces\РГРТУ\main.py"

.....

----------------------------------------------------------------------

Ran 5 tests in 0.000s

OK

[Done] exited with code=0 in 0.173 seconds

Вывод

В ходе работы мы получили знания об особенностях структурного тестирования программного обеспечения, а также умения и навыки его выполнения для оценки качества программного модуля, реализующего алгоритм поставленной задачи.