|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ |
| УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ |
| «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» |
|  |
| Институт информационных технологий и управления в технических системах |
| (полное название института) |
|  |
| кафедра «Информационные системы» |
| (полное название кафедры) |

**ОТЧЁТ**

по лабораторной работе №5

на тему«Исследование способов модульного тестирования программного обеспечения в среде NUnit»

по дисциплине **«**Тестирование программного обеспечения»

Вариант 22

Выполнил

студент группы ПИ/б-18-1-о

Маринин Игорь

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | | | | | | | |
|  |  | (должность, учёная степень преподавателя) | | | | | | | | |
|  |  |  | | | | | | | | |
|  |  | (ФИО преподавателя) | | | | | | | | |
|  |  | « |  | » |  |  |  | 20 | 20 | г. |
|  |  |  | | | | | | | | |
|  |  | (оценка) | | | | | | | | |
|  |  |  | | | | | | | | |
|  |  |  | | | | | | | | |

Севастополь 2020

* 1. **Цель работы**

Исследовать критические по времени выполнения участки программного кода и возможности их устранения. Приобрести практические навыки анализа программ с помощью профайлера NProfiler.

* 1. **Постановка задачи**
     1. Разработать программу на основе библиотеки классов, реализованной и протестированной в предыдущей работе. Программа должна как можно более полно использовать функциональность класса. При необходимости для наглядности профилирования в методы класса следует искусственно внести задержку выполнения.
     2. Выполнить профилирование разработанной программы, выявить функции, на выполнение которых тратится наибольшее время.
     3. Модифицировать программу с целью оптимизации времени выполнения.
     4. Выполнить повторное профилирование программы, сравнить новые результаты и полученные ранее, сделать выводы.
  2. **Ход работы**
     1. Разработает программу на языке Python с заведомо медленным алгоритмом.

**from** time **import** sleep

**class** **Multiply(object):**

'''

Этот класс организует возможность перемножения двух заданных значений типа int (x и y).

Их можно задать либо при создании объекта, либо обратившись к полям напрямую.

'''

x **=** **None**

y **=** **None**

**def** \_\_init\_\_**(**self**,** x**:int** **=** **None,** y**:int** **=** **None):**

self**.**x **=** x

self**.**y **=** y

**def** do**(**self**):**

'''

Этот метод умножает значение 'x' на значение 'y'

Результат: Результат перемножения переменых 'x' и 'y'

Возвращаемый тип: int

'''

# Проверка на существование значений

**if((**self**.**x **is** **None)** **or** **(**self**.**y **is** **None)):**

**raise** **ValueError(**"Пожалуйста, вначале укажите значения 'x' и 'y'"**)**

# Если одно из значений равно нулю, то и результат будет равен нулю

**if((**self**.**x **==** 0**)** **or** **(**self**.**y **==** 0**)):**

**return** 0

result **=** 0

# Запатентованная формула перемножения нескольких переменныйх типа int

**for** i **in** **range(**self**.**y**):**

result **+=** self**.**x

**return** result

**class** **RectangleArea(object):**

'''

Этот класс высчитывает площадь прямоугольника по заданным значениям (длины сторон) типа int (a и b).

Их можно задать либо при создании объекта, либо обратившись к полям напрямую

'''

a **=** **None**

b **=** **None**

**def** \_\_init\_\_**(**self**,** a**:int** **=** **None,** b**:int** **=** **None):**

self**.**a **=** a

self**.**b **=** b

**def** do**(**self**):**

'''

Этот метод высчитывает площадь прямоугольника по двум заданным значениям типа int (длинам сторон).

Результат: Площадь прямоугольника

Возвращаемый тип: int

'''

# Проверка на существование значений

**if((**self**.**a **is** **None)** **or** **(**self**.**b **is** **None)):**

**raise** **ValueError(**"Пожалуйста, вначале укажите длины сторон 'a' и 'b'"**)**

# Проверка на правильность ввода значений

**if(** **(**self**.**a **<** 0**)** **or** **(**self**.**b **<** 0**)** **):**

**raise** **ValueError(**"Введены неверные данные! Длины сторон не могут быть отрицательными"**)**

**elif** **(** **(**self**.**a **==** 0**)** **or** **(**self**.**b **==** 0**)** **):**

**raise** **ValueError(**"Введены неверные данные! Длины сторон не могут иметь нулевую длину"**)**

# Производим математическое вычисление площади прямоугольника по заданным значениям

patentedMultiply **=** Multiply**(**self**.**a**,** self**.**b**)**

**return** patentedMultiply**.**do**()**

* + 1. Выполним профилирование программы (рисунок 6.1).

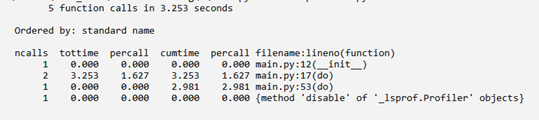


Рисунок 6.1 – Результаты профилирования программы

В ходе выполнения профилирования было выяснено, что узким местом программы является метод do класса Multiply, где реализован сомнительный алгоритм умножения.

* + 1. Модифицируем программу с целью оптимизации

**from** time **import** sleep

**class** **Multiply(object):**

'''

Этот класс организует возможность перемножения двух заданных значений типа int (x и y).

Их можно задать либо при создании объекта, либо обратившись к полям напрямую.

'''

x **=** **None**

y **=** **None**

**def** \_\_init\_\_**(**self**,** x**:int** **=** **None,** y**:int** **=** **None):**

self**.**x **=** x

self**.**y **=** y

**def** do**(**self**):**

'''

Этот метод умножает значение 'x' на значение 'y'

Результат: Результат перемножения переменых 'x' и 'y'

Возвращаемый тип: int

'''

# Проверка на существование значений

**if((**self**.**x **is** **None)** **or** **(**self**.**y **is** **None)):**

**raise** **ValueError(**"Пожалуйста, вначале укажите значения 'x' и 'y'"**)**

# Если одно из значений равно нулю, то и результат будет равен нулю

**if((**self**.**x **==** 0**)** **or** **(**self**.**y **==** 0**)):**

**return** 0

result **=** 0

## Запатентованная формула перемножения нескольких переменныйх типа int

#for i in range(self.y):

# result += self.x

# Осознаем, что патент с велосипедом сомнительный, и начнём использовать встроенные методы

result **=** self**.**x **\*** self**.**y

**return** result

**class** **RectangleArea(object):**

'''

Этот класс высчитывает площадь прямоугольника по заданным значениям (длины сторон) типа int (a и b).

Их можно задать либо при создании объекта, либо обратившись к полям напрямую

'''

a **=** **None**

b **=** **None**

**def** \_\_init\_\_**(**self**,** a**:int** **=** **None,** b**:int** **=** **None):**

self**.**a **=** a

self**.**b **=** b

**def** do**(**self**):**

'''

Этот метод высчитывает площадь прямоугольника по двум заданным значениям типа int (длинам сторон).

Результат: Площадь прямоугольника

Возвращаемый тип: int

'''

# Проверка на существование значений

**if((**self**.**a **is** **None)** **or** **(**self**.**b **is** **None)):**

**raise** **ValueError(**"Пожалуйста, вначале укажите длины сторон 'a' и 'b'"**)**

# Проверка на правильность ввода значений

**if(** **(**self**.**a **<** 0**)** **or** **(**self**.**b **<** 0**)** **):**

**raise** **ValueError(**"Введены неверные данные! Длины сторон не могут быть отрицательными"**)**

**elif** **(** **(**self**.**a **==** 0**)** **or** **(**self**.**b **==** 0**)** **):**

**raise** **ValueError(**"Введены неверные данные! Длины сторон не могут иметь нулевую длину"**)**

# Производим математическое вычисление площади прямоугольника по заданным значениям

patentedMultiply **=** Multiply**(**self**.**a**,** self**.**b**)**

**return** patentedMultiply**.**do**()**

* + 1. Повторим профилирование программы (рисунок 6.2).

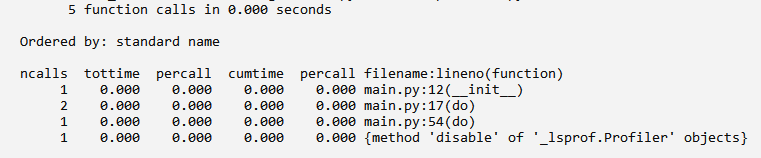


Рисунок 6.2 – Результаты повторного профилирования

В ходе повторного профилирования узкое место было устранено.

**Вывод**

В ходе данной лабораторной работы были изучены подходы профилированию программного обеспечения. В результате проведения профилирования было выявлено и устранено узкое место в программе. Кроме этого были приобретены практические навыки анализа программ с помощью профайлера cProfiler.