**ОПИСАНИЕ ЗАДАНИЯ:**

Разработать программный продукт с использованием динамической проверки типов во время выполнения. Программа должна содержать следующие структуры:

| **Обобщенный артефакт, используемый в задании** | **Базовые альтернативы** |
|---|---|
| Плоская геометрическая фигура, размещаемые в координатной сетке | 1. Круг<br>2. Прямоугольник<br>3. Треугольник |

Для всех альтернатив общей переменной является **цвет** (перечислимый тип). Он принимает значения: бесцветный (дефолтный), красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый. Общей функция всех альтернатив выступает вычисление площади фигуры (действительное число). В качестве дополнительной функции контейнера необходимо удалять из него те фигуры, площадь которых меньше, чем среднее арифметическое площадей всех фигур.

Также необходимо разработать тестовые входные данные и провести тестирование и отладку программы на этих данных, описать структуру используемой ВС с наложением на нее обобщенной схемы разработанной программы; зафиксировать количество заголовочных файлов, программных файлов, общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных. Реализована автогенерация тестов.

**ОСНОВНЫЕ ФАРАКТЕРИСТИКИ ПРОГРАММЫ:**

- Число модулей реализации – 9
- Общий размер исходных текстов – 490 строк вместе с комментариям.
- Время выполнения программы для различных тестовых прогонов:

| Номер теста | Время выполнения в секундах |
|---|---|
| 1 | 0.008934 |
| 2 | 0.009482 |
| 3 | 0.008678 |
| 4 | 0.005993 |
| 5 | 0.009463 |

**СРАВНЕНИЕ С ПРЕДЫДУЩИМИ ВЕРСИЯМИ ПРОГРАММЫ:**

Предыдущее задание было посвящено статической типизацией. Использование динамического связывания и виртуальной машины Python, вывод данных в консоль привели к ряду отличий данной версии программы, а именно:

Программа стала работать существенно медленнее предыдущих 2-х версий. Примерно в 5–10 раз по сравнению с аналогичной программой, разработанной на C++, и программой на C (процедурный подход). Связано это с наличием интерпретатора, а также со спецификой языка с динамической типизацией, так как нужно время на связывание.

Пропала возможность измерять размер типов. В Python нет возможности получить размер типа (только объекта какого-то типа), поэтому провести анализ затрачиваемой памяти нельзя.

Количество строк уменьшилось в два раза, то есть Python позволяет реализовывать логику работы программы в кратчайшие сроки, в качестве издержек - скорость работы программы падает.