Национальный исследовательский университет «Высшая школа экономики»

Домашнее задание №3

Архитектура вычислительных систем

Пояснительная записка

Исполнитель

студент группы БПИ202

Галюта Ева Валерьевна

2021 г.

Задание

Вариант: 46

Номер задачи: 4

Номер функции: 6

Составление программы по условию (Python):

* Обобщенный артефакт, используемый в задании: объемная (трехмерная) геометрическая фигура.
* Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив):

1. Шар (целочисленный радиус)
2. Параллелепипед (три целочисленных ребра)
3. Правильный тетраэдр (длина ребра – целое)

* Общие для всех альтернатив переменные:

1. Плотность материала фигуры (действительное число)

* Общие для всех альтернатив функции

1. Вычисление площади поверхности (действительное число)

* Расположение всех альтернатив в едином контейнере.
* Упорядочивание элементов контейнера по возрастанию используя Shaker Sort. В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.
* Формат ввода:

1. Готовые тестовые файлы (./figures\_hw2 -f <input file with data> <output file> <output sorted file>) input file:
2. Первый аргумент тип фигуры число от 1 до 3 (1 – сфера, 2 – параллелепипед, 3 – правильный тетраэдр).
3. Второй аргумент уникальный (сфера – радиус (целочисленное значение), параллелепипед – 3 стороны (целочисленные значения), правильный тетраэдр – сторона (целочисленное значение)).
4. Третий аргумент общий для всех – плотность материала фигуры – действительное число.
5. Генерация случайных значений (./figures\_hw2 -n <number of figures> <output file> <output sorted file>)

* [Новое] Использование динамической проверки типов во время выполнения программы.

Структурная схема изучаемой архитектуры ВС с размещенной на ней разработанной программы

Таблица типов (типы динамические, нельзя точно сказать размер переменной, в таблице указаны примерные типы данных (~))

|  |
| --- |
| class Sphere |
| \_\_radius : ~int  \_density : ~double (class Shape) |
| class Parallelepiped |
| \_\_a, \_\_b, \_\_c : ~int  \_density : ~double (class Shape) |
| class Tetrahedron |
| \_\_edge : ~int  \_density : ~double (class Shape) |
| сlass Container |
| \_\_list : ~Shape[30000] |
| class Shape |
| \_density : ~double |

Graphical user interface

Description automatically generated with low confidenceТаблица классов

Table

Description automatically generatedПамять программы (типы динамические, нельзя точно сказать размер переменной, в таблице указаны примерные типы данных (~))

Стек вызовов (| - вызов одной из функций, ? – поведение программы при вызове функции)

|  |
| --- |
| main  err\_message1? – ~~main~~  Container::input (error – ~~main~~) | Container::in\_rnd | ErrMessage2? – ~~main~~  ~~Container::input~~ | ~~Container::int\_rnd~~  Container::out  ~~Container::out~~  Container::shaker\_sort  ~~Container::shaker\_sort~~  Container::out  ~~Container::out~~  ~~main~~ |

Характеристики программы

Интерфейсные модули: 0

Модули реализации: 3

Общий размер исходных тестов: 496 Кб

Общий размер результатов тестов: 4.1 Мб

Размер исполняемого кода: 5.9912 Кб

Результаты тестов (в секундах)

|  |  |  |
| --- | --- | --- |
| Количество элементов | Ввод из файла | Генерация элементов |
| 20 (test\_1\_20.txt) | 0.01208 | 0.01639 |
| 100 (test\_2\_100.txt) | 0.02227 | 0.02146 |
| 1000 (test\_3\_1000.txt) | 0.64616 | 0.69038 |
| 10000 (test\_4\_10000.txt) | 62.01779 | 61.77551 |
| 20000 (test\_5\_20000.txt) | 237.04018 | 242.1274 |

Различия во времени не зависят от типа ввода, минимальные различия может выдавать сортировка, так как генерируемые данные могут быть частично упорядочены.

Заключение

Разработали программный продукт с использованием динамической проверки типов во время выполнения программы. Разработка на Python намного быстрее, нежели на C++ (размеры исполняемого кода уменьшились почти в 3 раза.), но время исполнения медленнее, что заметно на тестах. Причиной невысокой производительности Python является его динамическая природа и универсальность. Его можно использовать как инструмент для решения коротких и разнообразнейших задач. Сравнение с предыдущей версией кода, а также основания для вывода [Приложение 1].

Приложение 1  
  
Сравнение с предыдущей версией:

Результаты тестов (в секундах)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Предыдущее решение | | Текущее решение | |
| Количество элементов | Ввод из файла | Генерация элементов | Ввод из файла | Генерация элементов |
| 20 (test\_1\_20.txt) | 0.000867 | 0.000766 | 0.01208 | 0.01639 |
| 100 (test\_2\_100.txt) | 0.002346 | 0.001917 | 0.02227 | 0.02146 |
| 1000 (test\_3\_1000.txt) | 0.069392 | 0.055530 | 0.64616 | 0.69038 |
| 10000 (test\_4\_10000.txt) | 3.914274 | 3.829114 | 62.01779 | 61.77551 |
| 20000 (test\_5\_20000.txt) | 12.225400 | 11.842741 | 237.04018 | 242.1274 |
| Размер исполняемого кода | 13.925 Кб | | 5.9912 Кб | |