Архитектура вычислительных систем.

Задание 3. Вариант 12. Функция 19.

Выполнила: Словягина Анна Михайловна

Студентка 2-го курса программной инженерии

группа БПИ207

1 Описание задания

* Обобщенный артефакт, используемый в задании:
  + животные.
* Базовые альтернативы и их уникальные параметры:
  + рыбы:

место проживания - перечислимый тип (река, озеро, океан, море)

* + птицы:

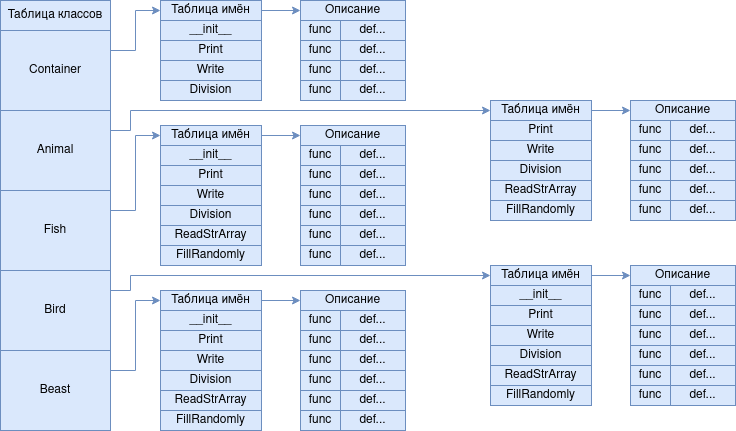
отношение к перелету - булевская величина (перелетные, не перелетные)

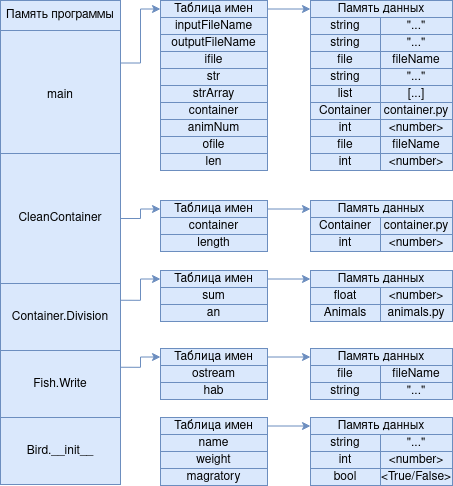
* + звери:

тип зверя - перечислимый тип (травоядные, хищники, насекомоядные)

* Общие для всех альтернатив переменные:
  + Название - строка символов;
  + Вес в граммах - целое число.
* Общие для всех альтернатив функции:
  + Частное от деления суммы кодов названия животного на вес - действительное число.
* Функция - удаление из контейнера тех элементов, для которых значение, полученное с использованием функции, общей для всех альтернатив, меньше, чем среднее арифметическое для всех элементов контейнера, полученное с использованием этой же функции.

2 Структурная схема изучаемой архитектуры ВС





3 Стек



4 Основные характеристики программы

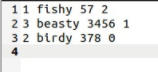
Число интерфейсных модулей: 0

Число модулей реализации: 10

Общий размер исходных текстов: 405

Формат ввода:

В каждой строке файла должно содержаться 4 переменные, разделенные пробелом:



Первая переменная отвечает за вид животного (1 – рыба, 2 – птица, 3 – зверь), вторая переменная – за название животного, третья переменная – за его вес в граммах, четвертая переменная – за уникальные параметры.

Уникальные параметры для рыбы – это место обитания:

1 – река

2 – озеро

3 – океан

4 – море

Уникальный параметр для птицы – переменная, которая отвечает за то, является ли птица перелетной

0 – нет

1 – да

Уникальные параметры для зверя – это тип животного

1 – травоядный

2 – хищный

3 – насекомоядный

Замечание:

При некорректном вводе типа животного, программа не добавляет данный элемент в контейнер и завершает ввод животных.

При некорректном вводе веса и уникального параметра для каждого животного, программа рандомно определяет этот параметр.

Расположение входных и выходных данных

Входные данные лежат в:

HomeWork\_3 /tests

Выходные данные лежат в:

HomeWork\_3/results

Ввод параметров:

Первый параметр влияет на формат ввода животных («-f» - ввод из файла, «-n» - рандомный ввод)

Если ввод производится из файла, следующие 2 параметра – входной и выходной файл (Пример: -f tests/inp1.txt results/res1.txt)

При рандомном заполнении следующие параметры – выходной файл, кол-во животных (Пример: -n results/resR1.txt 11)

5 Время выполнения программы для различных тестовых наборов данных

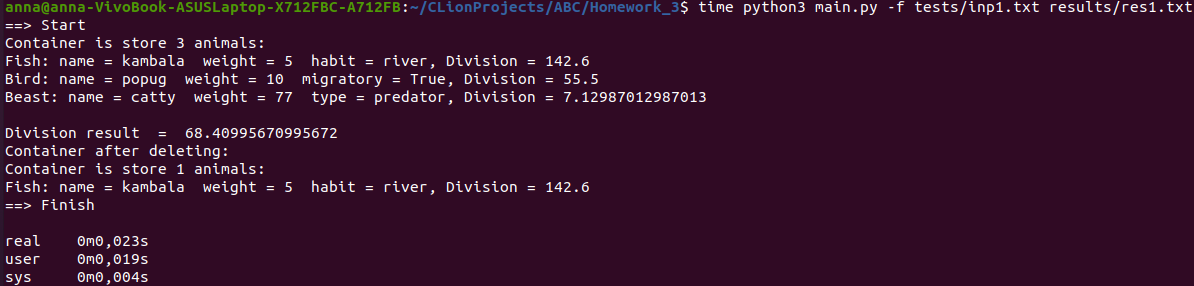
Ввод из файла:

1 тест:

Ввод:

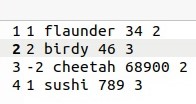


Запуск программы:

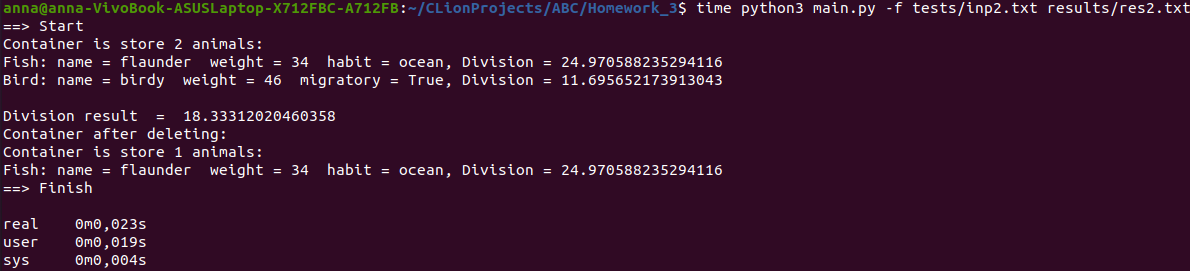


2 тест:

Ввод:



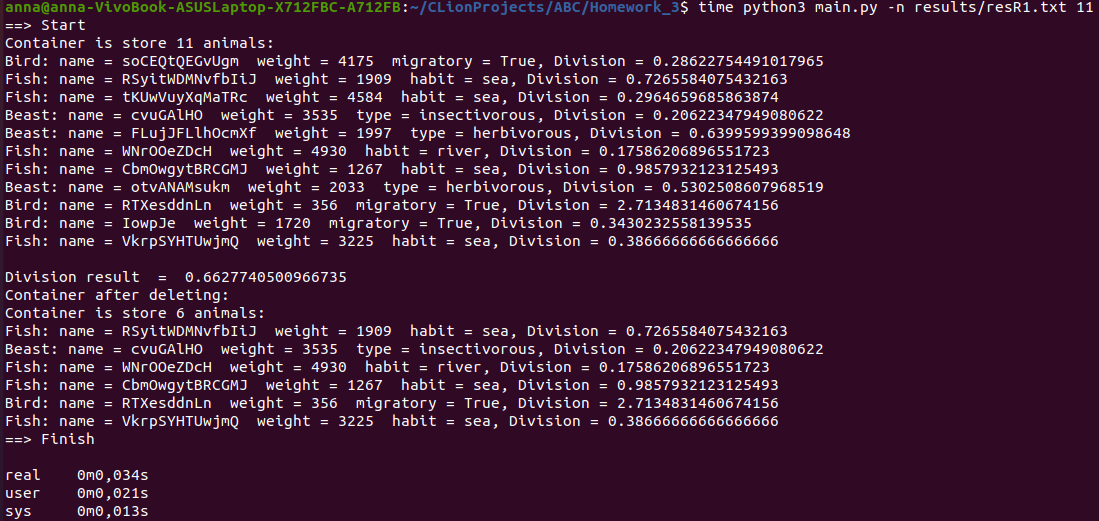
Запуск программы:



Рандомный ввод:

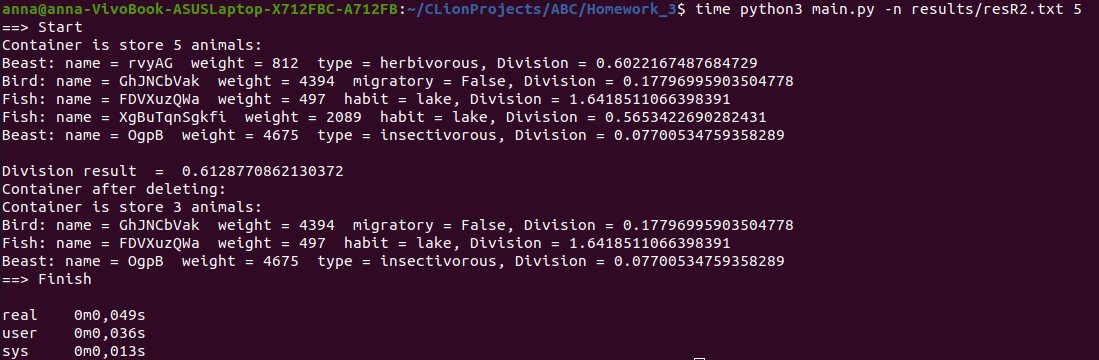
3 тест:

Запуск программы:



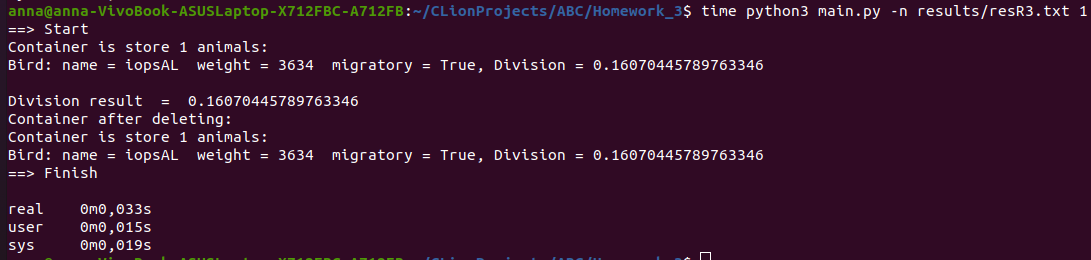
4 тест:

Запуск программы:



5 тест:

Запуск программы:



6 Сравнительный анализ

Основные характеристики 3 программы:

Число интерфейсных модулей: 0

Число модулей реализации: 10

Общий размер исходных текстов: 405

Основные характеристики 2 программы:

Число интерфейсных модулей: 5

Число модулей реализации: 6

Общий размер исходных текстов: 543

Основные характеристики 1 программы:

Число интерфейсных модулей: 5  
Число модулей реализации: 6  
Общий размер исходных текстов: 589

Сравнение времени исполнения программ:

|  |  |  |  |
| --- | --- | --- | --- |
| № | 1 программа | 2 программа | 3 программа |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |
| 5. |  |  |  |

Исходя из полученных данных, можно сделать вывод, что программа, написанная с помощью языка программирования использующего динамическую типизацию затрачивает в разы больше времени на выполнение кода, чем языки со статической типизацией. Это связано с тем, что языки программирования со статической типизацией определяют тип переменных на этапе компиляции а не во время работы программы. Однако, программа №3 занимает меньше строк кода, чем предыдущие 2 программы. Таким образом, можно сделать вывод, что в некоторых ситуация программы, использующие язык с динамической типизацией, проще и удобнее для написания, однако они требуют больше времени на выполнение, особенно сильно это будет заметно при работе с большим числом данных.