Сибирцева Анна
БПИ206

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Вариант: **222**
Задание: **12**
Функция: **16**

Описание задачи:

Необходимо реализовать программу, в которой базовым классом является животное, а его альтернативы (наследники): рыбы (поле – место проживания – перечислимый тип: река, озеро, море, пруд, океан), птицы(параметр – отношение к перелёту – перелётные или остающиеся на зимовку), звери(поле – тип питания – хищники, травоядные, насекомоядные, всеядные).

Общим для все классов являются: имя животного/его название – строка символов, вес животного в граммах – целое число.

Общей для всех классов функцией является нахождение частного от деления суммы кодов незашифрованной строки на вес, являющиеся действительным числом.

Обработка данных является их упорядочивание по убыванию, с использованием сортировки методом деления по пополам (*Binary Search*).

Основные характеристики программы:

Число интерфейсных модулей: **0**
Число модулей реализация: **6**
Общий размер исходных текстов: **~13,39 Кб**
Размер исполняемого файла (*main.py*): **1,59 Кб**
Время работы на тестах:

TEST 1: **0.002 с**
TEST 2: **0.008 с**
TEST 3: **0.001 с** *(*программа завершилась преждевременно из-за некорректных входных данных)*
TEST 4: **0.002 с**
TEST 5: **0.001 с** *(*программа завершилась преждевременно из-за некорректных входных данных)*
TEST 6: **5.862 с**
TEST 7: **0.003 с**
TEST 8: **0.002 с**

TYPE TABLE	
INT	14
FLOAT	16
STRING	35
CHAR[26]	132
ANIMAL LIST[]	28
class Beast: **__nutrition__: int**	**14** 14[0]
class Bird: **__is_migratory__: int**	**14** 14[0]
class Fish: **__habitat__: int**	**14** 14[0]
class Animal: **__nickname__: string** **__ weight__: int**	**49** 35[0] 14[35]

PROGRAM MEMORY	
main (): **argv:** **size: int** **all_animals: list** **input_stream: - _io.TextIOWrapper** **general_output: - _io.TextIOWrapper** **sorted_output: - _io.TextIOWrapper**	**450** 48[0] 14[48] 28[62] 120[90] 120[210] 120[330]
def quotient(self):: **chars_sum: float** **letter: char**	**42** 16[0] 26[16]
def input_random(size, all_animals): **i: int** **animal_type: int**	**28** 14[0] 14[14]
def input_from_file(size, all_animals, input_stream): **i: int** **animal_type: int** **parameters: list**	**104** 14[0] 14[14] 76[28]
def generate_name() **name_length: int** **name: string** **i: int**	**63** 14[0] 35[14] 14[49]

Дополнительная информация:

Программа была выполнена с использованием объектно-ориентированного подхода и динамической типизации. Разработка осуществлялась на языке программирования Python.

Подробная информация о корректном использовании находится в файле README.md

Ниже представлен ПРИБЛИЗИТЕЛЬНЫЙ стек вызовов:

На нём показаны только методы, которые вызываются прямиком из main.py, а так же методы сортировки. Всё остальное можно посмотреть в исходных файлах.

Сравнение программ:

Это третья программа выполненная по данному заданию. В этой версии основным языком программирования был Python. Написание кода допускалось в свободной форме, но были использованы ООП и динамическая типизация. Из отличий, которые появились в данной версии (сравнив три отчета) можно выделить следующее:

Во-первых, нет файла .exe т.к. можно запустить любой из исполняемых файлов. Получается, что назовём «недо.exe» файл main.py, который по сравнению с предыдущими версия весит всего пару килобайт. Да и само по себе количество исходных текстов стало гораздо меньше (кол-во строк кода сократилось почти вдвое)

Во-вторых, программа гораздо дольше сортирует объекты в массиве. Почти в три раза дольше чем предыдущие версии, что, скорее всего просто связано с тем, что Python довольно медленный язык по сравнению с предыдущими двумя.

В-третьих, в остальных тестах данная реализация (по времени) показала результаты лучше, чем обе предыдущие, так что можно сделать выводы, что шестой тест (который является исключением) показывает, что питон гораздо быстрее справляется с подобными ‘быстрыми запросами’,

но сильно отстает по времени при обработки большого количества данных.

Делая выводы из всего перечисленного выше, можно сказать, что данная реализация определено является более эффективной как по времени, так и по памяти для небольших входных данных, иначе ее использование просто будет не рационально, ведь первая реализация на C++ в стиле C все еще является абсолютным лидером в данном плане.