НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Архитектура вычислительных систем.
Пояснительная записка.
Практическое задание 3
Вариант 183 (Задача 1. Функция 14).

Исполнитель
Студент БПИ205
Никитин Никита Евгеньевич

Описание полученного задания

Разработать программу, состоящую из контейнера, способного хранить базовые альтернативы обобщенного артефакта.
Обобщенный артефакт — Плоская геометрическая фигура, размещаемая в координатной сетке.
Базовые альтернативы:
1. Круг (целочисленные координата центра окружности, радиус)
2. Прямоугольник (целочисленные координаты левого верхнего и правого нижнего углов)
3. Треугольник (целочисленные координаты трех углов)
Общая переменная для всех альтернатив - Цвет фигуры (перечислимый тип) = {красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый}
Общая для всех альтернатив функция: Вычисление площади фигуры (действительное число)
Контейнер должен уметь заполняться из файла и случайно (во входном файле нужно указать кол-во элементов для случайной генерации), выводить содержимое в файл(в удобном для пользовательского восприятия пользователем), сортироваться по общей для всех альтернатив функции методом Сортировки Шелла.
Примеры вариантов запуска программы:
Для чтения данных об объектах из файла:
command inputFile outputFile
Для случайной генерации объектов:
command -r inputFile outputFile

Структурная схема изучаемой архитектуры ВС

Отображение содержимого классов

Таблица классов	Таблица имен	Описание	
Container	__init__	func	def ...
	fill	func	def ...
	random_fill	func	def ...
	output_data	func	def ...
	sort	func	def ...
Point	__init__	func	def ...
	__str__	func	def ...
Colors	__str__	func	def ...
Figure	__init__	func	def ...
	input_figure	func	def ...
	random_input	func	def ...
	__str__	func	def ...
Circle	__init__	func	def ...
	input_figure	func	def ...
	random_input	func	def ...
	area	func	def ...
	__str__	func	def ...
Rectangle	__init__	func	def ...
	input_figure	func	def ...
	random_input	func	def ...
	area	func	def ...
	__str__	func	def ...
Triangle	__init__	func	def ...
	input_figure	func	def ...
	random_input	func	def ...
	area	func	def ...
	__str__	func	def ...

Отображение на память методов классов

Память программы	Таблица имен	Память
main.py	sys.argv	List[str]
	cont	Container
	input_file	file
	size	int
	output_filename	string
	output_file	file
	dataLines	List
Container.__init__	self.__figures	List
Container.fill	data	List
	type	int
Container.random_fill	size	int
	type	int
	color	int
Container.output_data	file	file
Container.sort	n	int
	gap	int
	i	int
	j	int
	temp	Circle/Rectangle/Triangle
Point.__init__	self.__x	int
	self.__y	int
Point.__str__	-	-
Colors.__str__	name	List
Figure.__init__	self.color	int
Figure.__str__	-	-
Circle.__init__	self.center	Point
	self.__radius	Point
Circle.input_figure	color	int
	center	Point
	radius	int
	Circle	Circle
Circle.random_input	color	int
	center	Point
	radius	int

	Circle	Circle
Circle.area	-	-
Circle.__str__	-	-
Rectangle.__init__	self.__left_top	Point
	self.__right_bottom	Point
Rectangle.input_figure	color	int
	left_top	Point
	right_bottom	Point
	Rectangle	Rectangle
Rectangle.random_input	color	int
	left_top	Point
	right_bottom	Point
Rectangle.area	-	-
Rectangle.__str__	-	-
Triangle.__init__	self.__p1	Point
	self.__p2	Point
	self.__p3	Point
Triangle.input_figure	color	int
	p1	Point
	p2	Point
	p3	Point
	Triangle	Triangle
Triangle.random_input	color	int
	p1	Point
	p2	Point
	p3	Point
	Triangle	Triangle
Triangle.area	-	-
Triangle.__str__	-	-

Общие характеристики программы

Число модулей реализации — 6

Общий размер исходных кодов — 248

Время работы программы на тестовых данных

input1.txt — 3 объекта
real 0m0,072s
user 0m0,059s
sys 0m0,012s

input2.txt — 6 объектов
real 0m0,087s
user 0m0,076s
sys 0m0,001s

input3.txt — 10 объектов
real 0m0,069s
user 0m0,047s
sys 0m0,019s

input4.txt — 14 объектов
real 0m0,068s
user 0m0,047s
sys 0m0,016s

input5.txt — 20 объектов
real 0m0,079s
user 0m0,050s
sys 0m0,026s

Время работы при рандомной генерации 1000 элементов
real 0m0,169s
user 0m0,131s
sys 0m0,015s

Время работы при рандомной генерации 10000 элементов
real 0m1,289s
user 0m1,183s
sys 0m0,064s

Сравнение с предыдущим методом реализации:

Таблица сравнения размера и времени исполнения программ

	Статическая типизация процедурный подход	Статическая типизация объектно-ориентированный подход	Динамическая типизация
Размер исходных кодов прораммы, строки	444	429	248
Считывание из файла 3 объектов, сек	0,003	0,004	0,072
Считывание из файла 20 объектов, сек	0,005	0,005	0,079
Случайная генерация 1000 объектов, сек	0,010	0,019	0,169
Случайная генерация 10000 объектов, сек	0,059	0,081	1,289

Программа с динамической типизацией на Python занимает меньше строк кода ввиду отсутствия заголовочных файлов. Также влияет тот факт, что Python язык более высокого уровня, и много из того, что в C/C++ приходится делать явно, в Python скрыто «под капотом». За это, к сожалению, приходится расплачиваться временем исполнения.

Программа на Python работает в 10-15 раз дольше, чем на C/C++. Это обусловлено тем, что программа на Python интерпретируется в runtime, а не компилируются и исполняются как в C/C++. Однако это позволяет не тратить время на предварительную компиляцию всего проекта. Так известны случаи, когда программисты уходили вечером с работы и ставили проект компилироваться на всю ночь.