Домашнее задание по предмету «Архитектура вычислительных систем» №3

Выполнила Шелемех Е.В, группа БПИ206

1. Описание полученного задания

Номер варианта – 226 -> условие задачи – 2, номер функции – 17

Условие задачи:

2. Плоская геометрическая фигура, размещаемая в координатной сетке	1. Круг (целочисленные координата центра окружности, радиус) 2. Прямоугольник (целочисленные координаты левого верхнего и правого нижнего углов) 3. Треугольник (целочисленные координаты трех углов)	Цвет фигуры (перечислимый тип) = {красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый}	Вычисление периметра фигуры (действительное число)

Функция обработки данных в контейнере:

17. Упорядочить элементы контейнера по убыванию используя сортировку с помощью разделения (Quick Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Описание работы программы

Запуск программы осуществляется из командной строки, в которой указываются: имя запускаемой программы; имя файла с исходными данными; имя файла с выходными данными. В файле с исходными данными в первой строке должен находится тип ввода данных: “random” или “file”. Если тип ввода - “random”, то во второй строке должно быть положительное число менее 10000, т.е кол-во генерируемых фигур. Если тип ввода - “file”, то в последующих строках вводится информация о фигурах в формате:

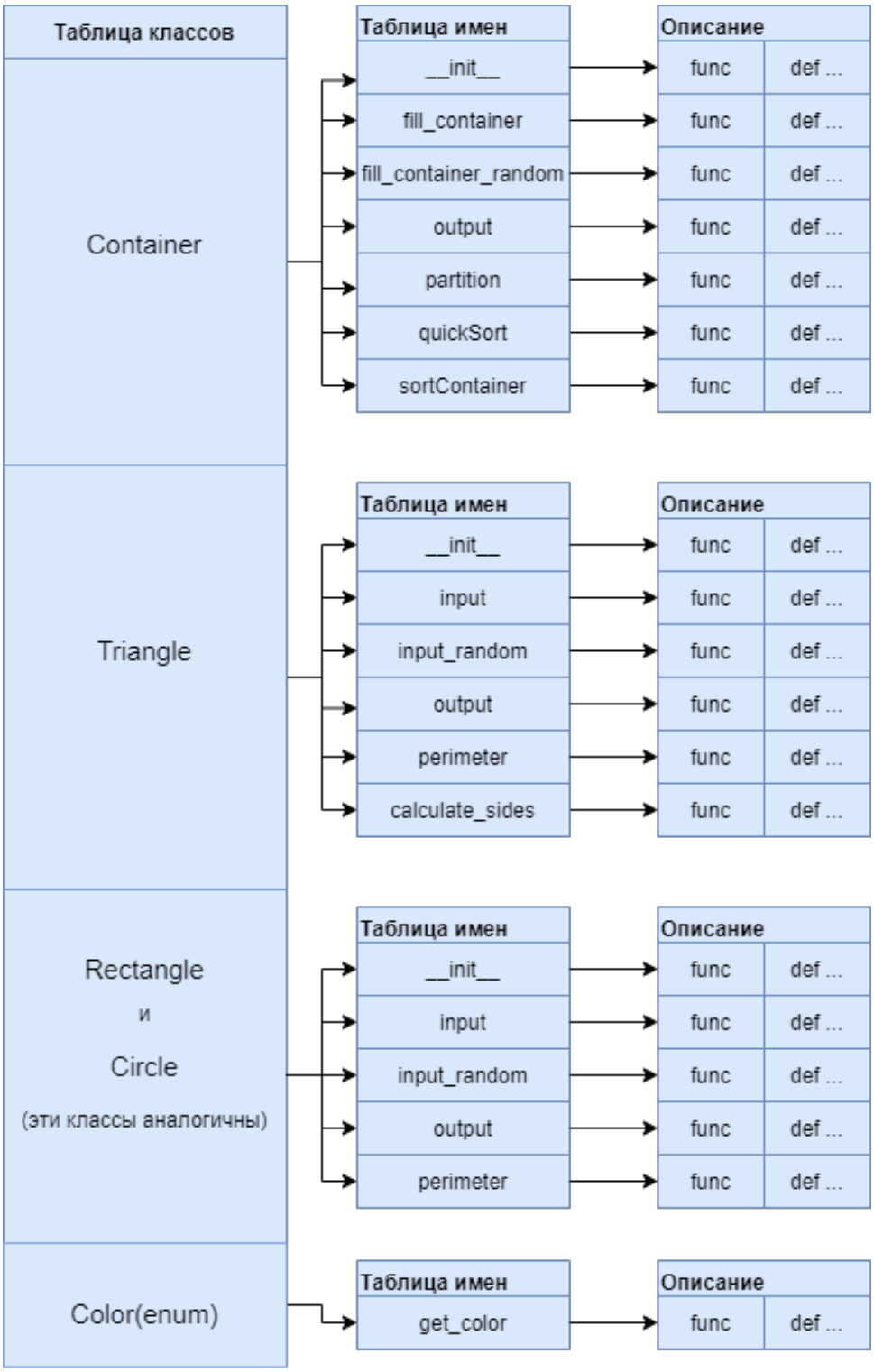
1. тип_фигуры (целое число от 1 до 3 включительно)
2. данные о фигуре
   2.1 если треугольник: цвет a_x a_y b_x b_y c_x c_y
   2.2 если прямоугольник: цвет a_x a_y b_x b_y
   2.3 если круг: цвет center_x center_y radius

цвет – целое от 0 до 6; a_x, a_y, b_x, b_y, c_x, c_y, center_x, center_y, radius – целые числа;

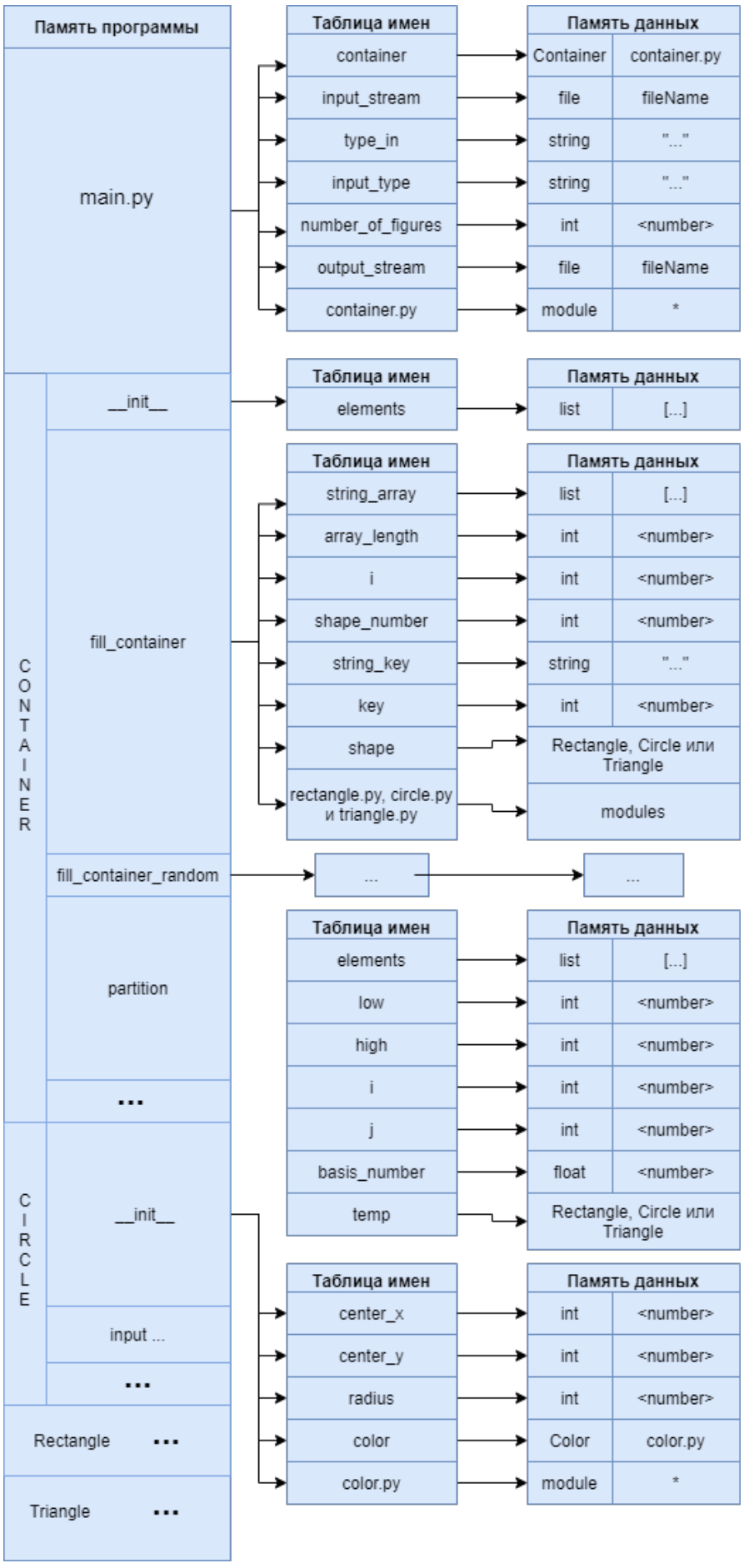
Результат работы программы, т.е вывод элементов контейнера до сортировки и вывод элементов контейнера после сортировки, помещается в выходной файл, указанный пользователем в качестве аргумента при запуске программы.

2. Архитектура

Отображение содержимого классов

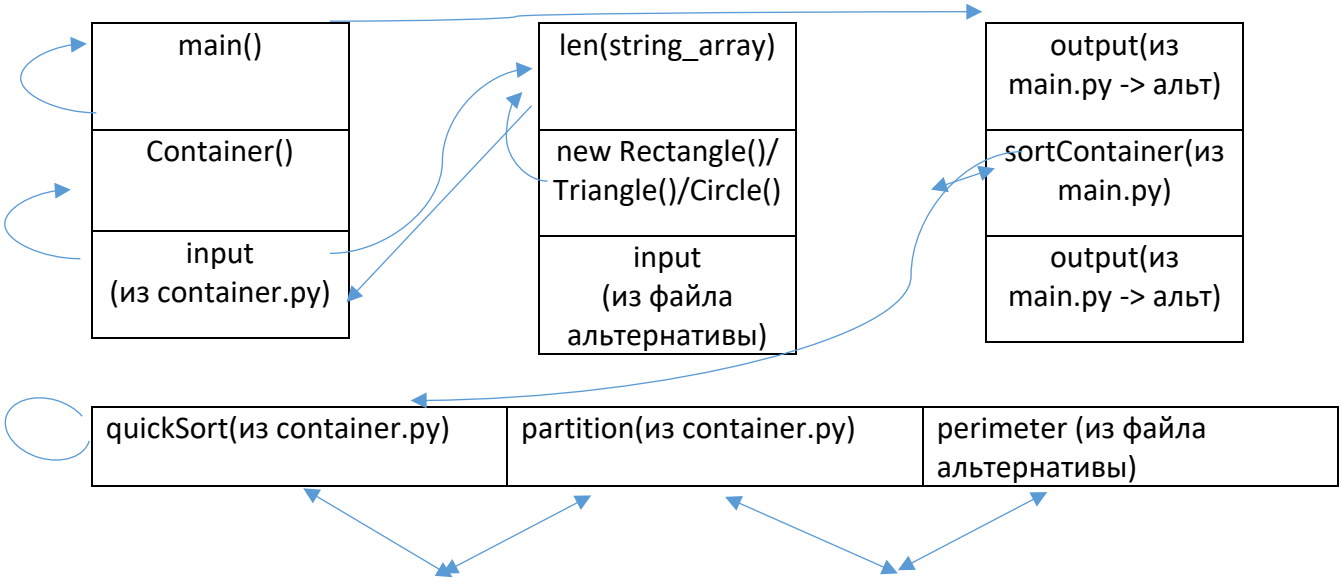


Отображение на память некоторых методов классов

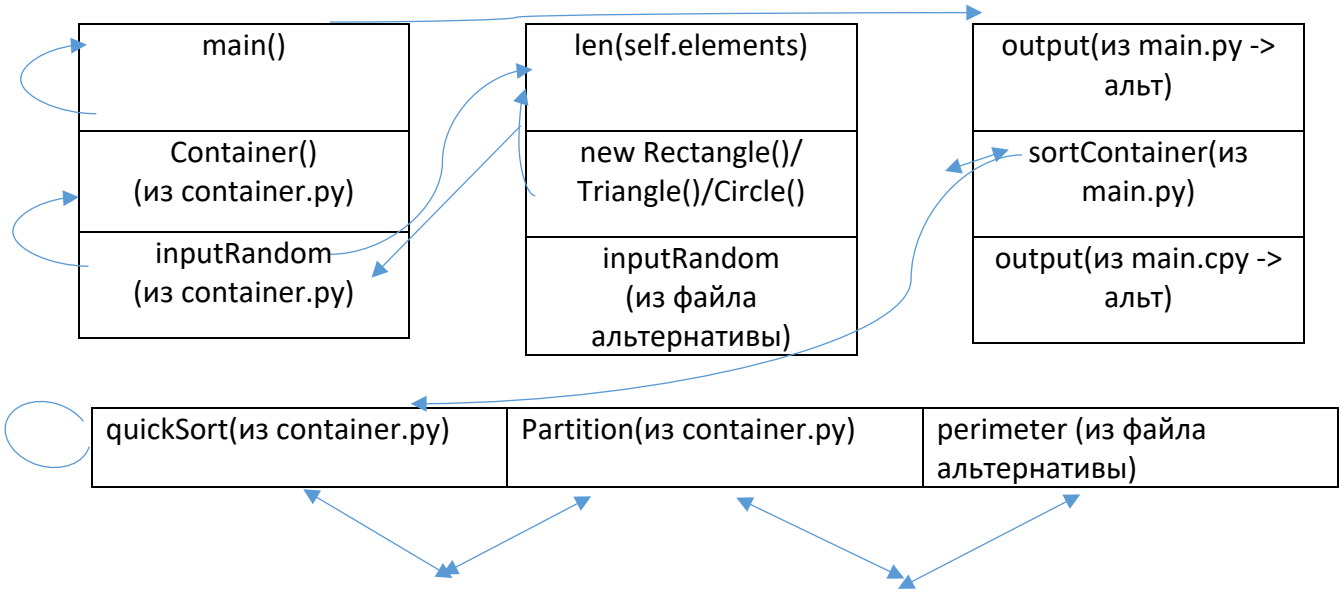


Стек вызовов

1. *Данные вводятся из файла*



2. *Данные вводятся рандомно*



3. **Характеристики**

Количество заголовочных файлов = 0

Количество модулей реализации = 6

Общий размер исходных текстов программы = 10.3 Кб

Время выполнения программы(на приложенных тестовых наборах):

Входной файл	Тип ввода данных	Описание тестового набора	Выходной файл	Время выполнения		
				процед. пар-ма + стат.тип.	ООП + стат.тип	динам. тип.
tests\test1.txt	Из файла	8 элементов	tests\output_test1.txt	9мс	3мс	1.2мс
tests\test2.txt	Из файла	11 элементов	tests\output_test2.txt	5мс	4мс	1.3мс
tests\test3.txt	рандом	66 ранд. элементов	tests\output_test3.txt	7мс	4мс	4мс
tests\test4.txt	рандом	Некорректное число элементов	tests\output_test4.txt	3мс	3мс	1мс
tests\test5.txt	Некорректный тип ввода	-	tests\output_test5.txt	1мс	3мс	1мс
tests\test6.txt	рандом	1000 ранд. элементов	tests\output_test6.txt	15мс	11мс	62мс
tests\test7.txt	рандом	10000 эл-ов	tests\output_test7.txt	173мс	142мс	705мс

4. Сравнение

Отличия от процедурного и ООП подходов:

1. Размер исходных текстов программы значительно уменьшился в сравнении с процедурным и ООП подходами.
2. Время выполнения на маленьких объемах данных уменьшилось, но при вводе большого кол-ва элементов программа с динамической типизацией на python работает значительно медленнее, чем программы с другими подходами.
3. Динамическая типизация позволяет объявлять переменные без указания их типа.
4. Динамическая типизация значительно сокращает время написания программы.
5. Отсутствие явно указанного типа на этапе компиляции повышает вероятность ошибок во время исполнения.