ОТЧЕТ ПО ПРОГРАММЕ, РАЗРАБОТАННОЙ НА СИ.

ВАРИАНТ 2, ФУНКЦИЯ 10

УСЛОВИЕ ЗАДАНИЯ:

Обобщённый артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив переменные	Общие для всех альтернатив функции
2. Плоская геометрическая фигура, размещаемые 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, жёлтый, зелёный, голубой, синий, фиолетовый}	Вычисление периметра фигуры (действительное число)

Функция:

10. Упорядочить элементы контейнера по убыванию используя сортировку с помощью прямого включения (Straight Insertion). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

СТРУКТУРА ПРОЕКТА:

project/ |

+-bin/ # собранный файл

+-container/ # файлы, описывающие класс Container

+-rnd/ # файлы, описывающие класс Random

+-triangle/ # файлы, описывающие класс Triangle

+-circle/ # файлы, описывающие класс Circle

+-rectangle/ # файлы описывают класс Rectangle

+-shape/ # файлы описывают класс Shape

+-tests_in/ # файлы с входными тестами

+-tests_out/ # результаты с тестовыми прогонами программы на тестах из папки tests_in

+-main.cpp # точка входа в программу

Спецификация ВС:
Operating System: macOS Big Sur 11.5.2

Спецификация средств разработки IDE:

Visual Studio Code (Version: 1.60.1 (Universal))

БИБЛИОТЕКИ

string.h time.h cstdlib cmath ctime
Средство сборки: CMake (v3.21.2)

ХАРАКТЕРИСТИКИ ПРОЕКТА

Количество заголовочных файлов (.h): 6

Количество программных объектов: 5

Общий размер исходных текстов: 24 К

Полученный размер исполняемого кода: 73 К

ВРЕМЯ РАБОТЫ ПРОГРАММЫ НА РАЗЛИЧНЫХ ТЕСТОВЫХ ПРОГОНАХ

	test1.txt	test2.txt	test3.txt	test4.txt	test5.txt
Time(seconds)	0.000356	0.000401	0.014865	0.207818	0.775573

Сравним его с характеристиками прошлой программы при одинаковом количестве элементов в тестовых наборах. Вот они

Для предыдущей программы имеем:

	test1.txt	test2.txt	test3.txt	test4.txt	test5.txt
Time(seconds)	0.000479	0.000510	0.017322	0.248745	0.913324

Мы видим, что на каждом из прогонов время работы новой программы меньше, хоть и незначительно. Поэтому я полагаю, что объектно-ориентированный подход немного эффективнее в рамках исполнения данной задачи.

ЗАПУСК ПРОГРАММЫ

В командной строке: ***./main input_file output_file***

Для рандомной генерации тестового набора данных: ***./main input_file output_file num***

(в input_file будет помещен сгенерированный тестовый набор, в output_file выходные данные для данного тестового набора, num – заданное число элементов тестового набора)