Домашнее задание по предмету «Архитектура вычислительных систем» №2

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
3. может быть пустая строка между вводом разных фигур, но не обязательно

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

Используемые структуры, переменные и их типы

int	4
double	8
class Rectangle a_x: int a_y: int b_x: int b_y: int наследование от Shape + память занимается до числа байт, кратного 8	32 4[0] 4[4] 4[8] 4[12] +12
class Triangle a_x: int a_y: int b_x: int b_y: int c_x: int	40 4[0] 4[4] 4[8] 4[12] 4[16]

c_y: int наследование от Shape + память занимается до числа байт, кратного 8	4[20] +12
class Circle center_x: int center_y: int radius: int наследование от Shape	24 4[0] 4[4] 4[8] +12
class Shape color: int указатель на таблицу виртуальных методов	12 4[0] 8[4]
class Container length: int Shape*elements[10000]	80008 4[0] 8*10000+4+4 = 80008

Глобальная память

В глобальной памяти нет переменных, следовательно, она пуста.

Локальная память

1. main()	
container: Container	80008[0]
input_stream: ifstream	520[80008]
input_type: string	32[80528]
number_of_figures: int	4[80560]
stream_out: ofstream	512[80564]
2. perimeter(…)	
из triangle.cpp	
ab: double	8[0]
ac: double	8[8]
bc: double	8[16]

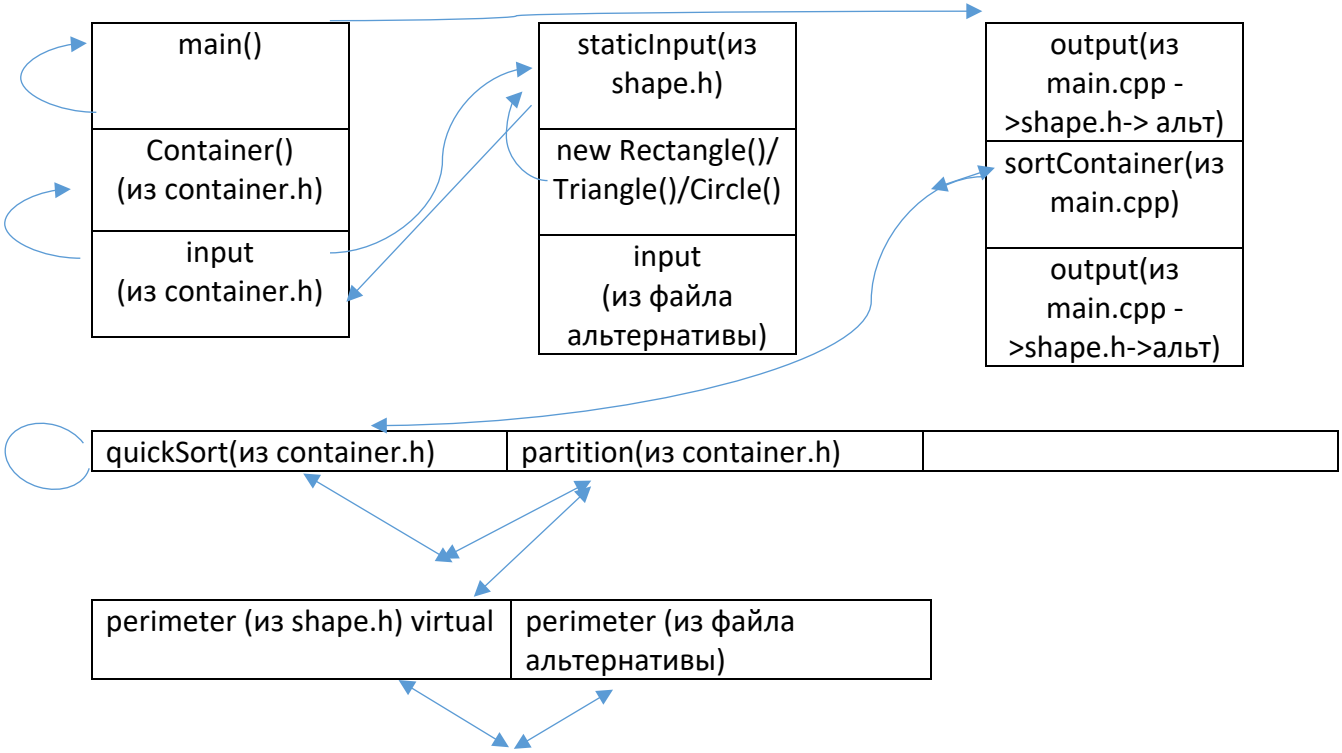
3. staticInput	
из shape.cpp	
k: int	4[0]
shape: Shape*	8[4]
4. partition(…)	
из container.cpp	
low: int	4[0]
high: int	4[4]
basis_number: double	8[8]
i: int	4[16]
j: int	4[20]

Куча

В куче хранятся элементы массива container: Shape*[10000] – указатели типа Shape, так как они были созданы динамически с помощью new. Каждый элемент занимает 8 байт.

Стек вызовов

1. *Данные вводятся из файла*



2. *Данные вводятся рандомно*

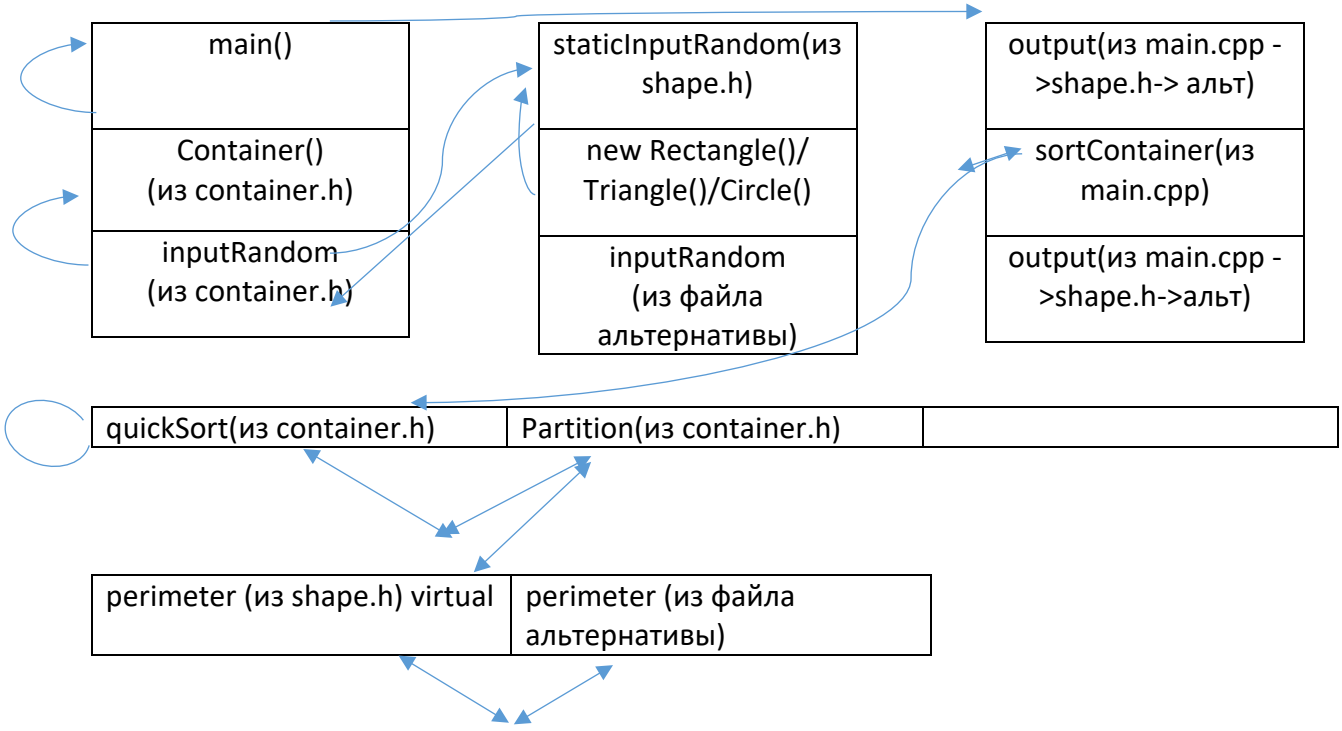
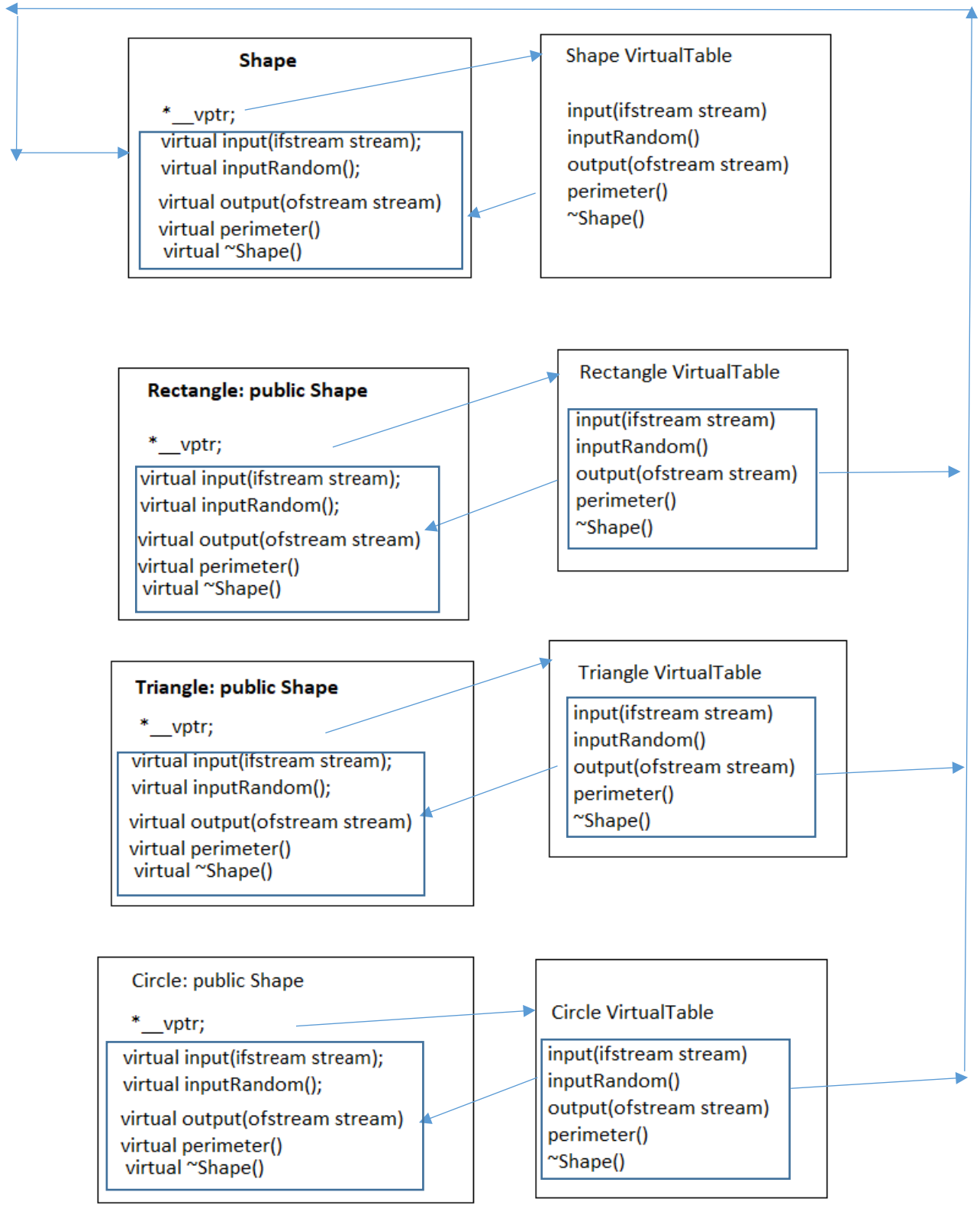


Таблица виртуальных методов



3. Характеристики

Количество заголовочных файлов = 6

Количество модулей реализации = 6

Общий размер исходных текстов программы = 14.7 Кб

Размер исполняемого файла = 789 Кб

Время выполнения программы(на приложенных тестовых наборах):

Входной файл	Тип ввода данных	Описание тестового набора	Выходной файл	Время выполнения
tests\test1.txt	Из файла	8 элементов	tests\output_test1.txt	3мс
tests\test2.txt	Из файла	11 элементов	tests\output_test2.txt	4мс
tests\test3.txt	рандом	66 ранд. элементов	tests\output_test3.txt	4мс
tests\test4.txt	рандом	Некорректное число элементов	tests\output_test4.txt	3мс
tests\test5.txt	Некорректный тип ввода	-	tests\output_test5.txt	3мс
tests\test6.txt	рандом	1000 ранд. элементов	tests\output_test6.txt	11мс

4. Сравнение

Отличия от процедурного подхода:

1. Для хранения типов необходимо больше памяти, чем в процедурном подходе, так как необходимо хранить ссылку на таблицу виртуальных методов.
2. Размер исходных текстов программы уменьшился.
3. Во время вызовов методов объектов альтернатив требуется меньший объем локальной памяти, так как не нужно передавать в метод сам объект, который на время выполнения функции занимает место в локальной памяти.
4. Время выполнения в среднем уменьшилось
5. Появилась таблица виртуальных методов

Сходства с процедурным подходом:

1. Практически не изменился стек вызовов функций
2. Переменные, хранящиеся в куче, не изменились
3. Объектно-ориентированный подход позволяет реализовать инкапсуляцию, наследование и полиморфизм