**Лабораторная работа №11. ООП. Геометрические фигуры**

**Костин Андрей Константинович, 3530904/20001, 5 вариант**

**Общая постановка задачи:**

1. Создать файл base-types.h, содержащий определения следующих структур:

* point\_t, представляющую собой точку на плоскости. Координаты должны храниться в полях x и y.
* rectangle\_t, описывающую прямоугольник шириной width и высотой height с центром в точке pos.

1. Создать файл shape.h, содержащий определение абстрактного класса Shape. Этот класс должен предоставлять следующие методы:

* getArea вычисление площади
* getFrameRect получение ограничивающего прямоугольника для фигуры (см. типы из предыдущего пункта), стороны ограничивающего прямоугольника всегда параллельны осям
* move перемещение центра фигуры, реализовать в двух вариантах:
  + перемещение в конкретную точку
  + смещение по осям абсцисс и ординат
* scale изотропное масштабирование фигуры относительно её центра с указанным коэффициентом
* getName получение названия фигуры (RECTANGLE, CIRCLE, COMPLEX и т.д.), метод использовать при выводе
* clone создание копии фигуры в динамической памяти и получение указателя на эту копию

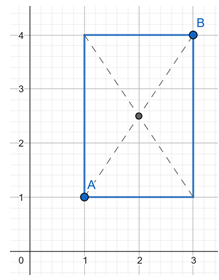
1. Реализовать класс Rectangle, производный от Shape, в файлах rectangle.h и rectangle.cpp, соответственно (см. раздел «Фигуры»).
2. Реализовать фигуру, указанную преподавателем (см. раздел «Фигуры»).
3. Реализовать класс CompositeShape, производный от Shape (см. раздел «Фигуры»), представляющий собой составную фигуру и хранящий массив в динамической памяти из произвольных фигур, реализованных в вашей программе. Использование умных указателей и контейнеров стандартной библиотеки недопустимо. Объекты класса должны корректно копироваться и перемещаться. При добавлении базовой (прямоугольник, круг и т.п.) фигуры в составную, добавляться должна копия базовой фигуры.
4. Перегрузить оператор вывода << для класса Shape. Вывод должен содержать в одну строчку: название фигуры, её площадь, координаты левого нижнего угла и правого верхнего угла ограничивающего прямоугольника. Элементы при выводе должны округляться до одного знака после запятой в соответствии с правилами математики.
5. Перегрузить оператор сравнения < для класса Shape. Сравнение проводить по площади фигуры.
6. Реализовать функцию, сортирующую массив указателей на фигуры в порядке неубывания их площадей.
7. Написать программу, которая считывает описания фигур и команд из файла (см. раздел «Требования») и после окончания ввода фигур, а также после выполнения каждой команды выводит в стандартный вывод на отдельных строках информацию о каждой фигуре в порядке неубывания их площадей. Вывод для составной фигуры должен быть таким же, как и для всех остальных: название, площадь, координаты углов ограничивающего прямоугольника.
8. Отдельно продемонстрировать работу всех методов копирования и перемещения составной фигуры.

**Фигуры**

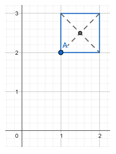
Каждая фигура описывается своим набором параметров. Отсутствие самопересечений, выпуклость фигур и корректное количество (и только количество) параметров гарантируется, если не сказано иного:

1. **Прямоугольник**. Описывается парой координат своих углов: левым нижним и правым верхним. Считается, что стороны прямоугольника параллельны осям координат. Центром фигуры считается точка пересечения диагоналей

RECTANGLE 1.0 1.0 3.0 4.0



1. **Квадрат**. Описывается координатами своего левого нижнего угла и длиной стороны. Считается, что стороны квадрата параллельны осям координат. Центром фигуры считается точка пересечения диагоналей  
   SQUARE 1.0 2.0 1.0



## Команды

Количество и корректность параметров гарантируются.

1. **Перемещение**. Выполняет смещение по осям абсцисс и ординат соответственно  
   MOVE 2.0 3.0
2. **Масштабирование**. Выполняет изотропное масштабирование относительно указанного центра. Первые два параметра – координаты центра, третий – коэффициент масштабирования. Коэффициент должен быть положительным.

SCALE 8.0 -3.0 2.0

**Детальные требования и тест план:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Требование** | **Детальные требования** | | **Тест план** | **Ожидаемый результат** | |
| 1. Данные должны быть заданы корректно | | | | | |
| * 1. Проверка на существование заданного файла | | Если условия не выполняются, сообщение: «Файл не был открыт. Первая часть программы не была завершена успешно.» | Верно: ifstream in("D:/Рабочий стол/in.txt"); | | Сообщение: «Файл не был открыт. Первая часть программы не была завершена успешно.» |
| Неверно: ifstream in("D:/Рабочий стол/ilk.txt"); | |
| * 1. Длина сторон должна быть положительной | | Если условия не выполняются, сообщение: «Длина стороны квадрата должна быть положительной.» | SQUARE 2.0 3.0 4.0 | | Сообщение: «Длина стороны квадрата должна быть положительной.» |
| SQUARE 2.0 3.0 -4.0 | |
| * 1. Максимальное количество фигур в составной должно быть положительно | | Если условия не выполняются, сообщение: «Максимальное кол-во фигур в составной фигуре должно быть положительным.» | COMPLEX -5 | | Сообщение: «Максимальное кол-во фигур в составной фигуре должно быть положительным.» |
| * 1. Максимальное количество фигур не достигнуто | | Если количество превышено, остальные фигуры игнорируются в составной и сообщение: «Достигнуто максимальное количество фигур составной фигуры, остальные значения не будут записаны.» | COMPLEX 2  RECTANGLE 6.0 7.0 9.0 8.0  SQUARE 9.0 -3.0 2.0  RECTANGLE -1.0 2.0 4.0 6.0  COMPLEXEND | | Сообщение: «Достигнуто максимальное количество фигур составной фигуры, остальные значения не будут записаны.» |
| * 1. Коэффициент масштабирование должен быть положительным | | Если условия не выполняются, сообщение: «Коэффициент масштабирования должен быть задан положительным числом.» | SCALE -1 -2 -2 | | Сообщение: «Коэффициент масштабирования должен быть задан положительным числом.» |

**Программа:** см. в Приложении к архиву.