**Задание №3  
Использование структур в C#**

В задачах этого семинара все объекты необходимо создавать в виде типа данных **struct**, принадлежащие типу-значению. Каждый новый тип данных **struct** должен использовать, при возможности, свойства более простого типа данных **struct**. Например, структура Line должна иметь 2 поля структуры Point. В методе **Main()** создать пример (демонстрация) использования созданных структур.

1. Создать структуру Point, описывающей точку на плоскости, заданную декартовыми координатами. Среди методов Point предусмотреть метод Distance, вычисляющий расстояние до заданной точки.
2. Создать структуру Line, описывающей линию на плоскости. Среди методов Line предусмотреть метод, определяющий принадлежность заданной точки линии. Реализовать метод, вычисляющий и выводящий на экран длину созданной линии.
3. Создать структуру Square, описывающей квадрат на плоскости со сторонами, параллельными осям координат. Среди методов Square предусмотреть метод, определяющий принадлежность заданной точки квадрату. Реализовать метод, вычисляющий и выводящий на экран площадь и периметр квадрата.
4. Создать структуру  Circle, описывающей круг на плоскости. Среди методов Circle предусмотреть метод, определяющий принадлежность заданной точки кругу. Реализовать метод, вычисляющий и выводящий на экран площадь и периметр круга.
5. Создать структуру  Rectangle, описывающей прямоугольник на плоскости со сторонами, параллельными осям координат. Среди методов Rectangle предусмотреть метод, определяющий принадлежность заданной точки прямоугольнику. Реализовать метод, вычисляющий и выводящий на экран площадь и периметр прямоугольника.
6. Создать структуру Rhomb, описывающей ромб на плоскости с осями, параллельными осям координат. Среди методов Rhomb предусмотреть метод, определяющий принадлежность заданной точки ромбу. Реализовать метод, вычисляющий и выводящий на экран площадь и периметр ромба.

**Использование перечислений в C#**

Написать программу управления конвейерной лентой на фабрике:

* Создать структуру **ConveyerControl**.
* Один из членов структуры - перечисление **action**, в котором определены принимаемые команды управления лентой - старт, стоп, вперед, назад.
* Еще один член структуры - метод **conveyer()**, принимающий единственный параметр типа **action**. Метод **conveyer()** при помощи инструкции **switch** определяет соответствующую команду и выводит в консоль состояние конвейерной ленты - "Запуск", "Остановка", "Перемещение вперед", "Перемещение назад".
* Все члены структуры имеют спецификатор доступа - **public**. В методе **Main()** создать пример (демонстрация) использования структуры **ConveyerControl**.
* Реализовать управление конвейерной лентой при помощи стрелок на клавиатуре: "Left" - "Остановка", "Right" - "Запуск", "Up" - "Перемещение вперёд", "Down" - "Перемещение назад". При нажатии клавиш ← → ↑ ↓ на экран должны выводиться соответствующие информационные сообщения о состоянии конвейерной ленты.