Лабораторная работа №11.
Классы. Конструкторы и деструкторы класса.
Компоненты класса. Перегрузка операций.

Разработать классы для описанных ниже объектов. Определить конструктор по умолчанию (без параметров), конструкторы с параметрами, конструктор копирования, деструктор. Определить методы класса set(…) для инициализации полей (данных) класса, get(…) для получения значений полей (данных) класса, show() для вывода данных на экран, другие методы согласно заданию. Обеспечить выбор методов и операций класса для пользователя.

Вариант 1.

Объект: **Circle**(окружность).
Данные: центр окружности (**double** x,y), радиус окружности (**double** r).
Методы:

- площадь круга, ограниченного окружностью (**double** Area (**void**));
- площадь сектора (**double** Sector (**double** x1, **double** y1, **double** x2, **double** y2), где точки с координатами (x1,y1), (x2,y2) лежат на окружности);
- площадь сегмента (**double** Segment (**double** x1, **double** y1, **double** x2, **double** y2), где точки с координатами (x1,y1), (x2,y2) лежат на окружности);
- метод, определяющий лежит ли точка с координатами (x,y) на окружности (**int** CirclePoint (**double** x, **double** y));
- метод, определяющий попадает ли точка внутрь круга, ограниченного данной окружностью (**int** Point(**double** x, **double** y));
- метод, который возвращает строку с уравнением касательной в точке (x,y) (**string** Tangent (**double** x, **double** y)).

Операторы:

- < (возвращает 1, если окружность слева целиком «лежит внутри» окружности справа, 0- в остальных случаях),
- > (возвращает 1, если окружность слева «целиком содержит» окружность справа, 0- в остальных случаях),
- = (возвращает 1, если площади соответствующих кругов равны);
- * («увеличивает» окружность в N раз);
- + (возвращает сумму площадей соответствующих кругов);
- - (возвращает модуль разности площадей соответствующих кругов).

Вариант 2.

Объект: **Triangle**(треугольник):
Данные: 3 точки с координатами (**double** x,y).
Методы:

- метод, определяющий, что данный треугольник существует (3 точки не лежат на одной прямой) (**int** TriangleRule(**void**));
- стороны треугольника (**double**[3] Leg(**void**));
- периметр треугольника (**double** Perimeter(**void**));
- площадь треугольника (**double** Area(**void**));
- метод, определяющий, что треугольник – прямоугольный (**int** Pithagor(**void**));
- радиус описанной окружности (**double** Radius (**void**));
- радиус вписанной окружности (**double** radius (**void**));
- угол треугольника напротив большей стороны (**double** Angle(**void**));
- метод, который возвращает строку с уравнением прямой, на которой лежит меньшая сторона (**string** Tangent(**void**));

Операторы:

- < (возвращает 1, если площадь треугольника слева меньше площади треугольника справа, 0- в остальных случаях);
- > (возвращает 1, если площадь треугольника слева больше площади треугольника справа, 0- в остальных случаях);
- = (возвращает 1, если площади соответствующих треугольников равны);
- * («растягивает» треугольник в N раз, относительно центра масс);
- || (совершает параллельный перенос треугольника на вектор (x,y));
- + (возвращает сумму площадей соответствующих треугольников);
- - (возвращает модуль разности площадей соответствующих треугольников).

Вариант 3.

Объект: **Square**(квадрат):

Данные: 2 точки с координатами противолежащих вершин (**double** x,y).

Методы:

- координаты вершин квадрата (**double**[4][2] Apex(**void**));
- сторона квадрата (**double** Leg(**void**));
- длина диагонали (**double** Diagonal(**void**));
- периметр квадрата (**double** Perimeter(**void**));
- площадь квадрата (**double** Area(**void**));
- радиус описанной окружности (**double** Radius (**void**));
- радиус вписанной окружности (**double** radius (**void**));
- метод, который совершает поворот квадрата относительно точки с координатами (x,y) на угол α (Square Turn (**double** x,**double** y,**double** alfa);
- метод, который возвращает строку с уравнением прямой, на которой лежит диагональ (**string** Tangent(**void**)).

Операторы:

- < (возвращает 1, если площадь квадрата слева меньше площади квадрата справа, 0 - в остальных случаях),
- > (возвращает 1, если площадь квадрата слева больше площади квадрата справа, 0 - в остальных случаях),
- = (возвращает 1, если площади соответствующих квадратов равны);
- * («растягивает» квадрат в N раз, относительно центра);
- || (совершает параллельный перенос квадрата на вектор (x,y))
- + (возвращает сумму площадей соответствующих квадратов);
- - (возвращает модуль разности площадей соответствующих квадратов).

Вариант 4.

Объект: **Segment** (отрезок):

Данные: 2 точки с координатами концов отрезка (**double** x,y).

Методы:

- длина отрезка (**double** Length(**void**));
- метод, возвращающий тангенс угла наклона данного отрезка к положительному направлению оси OX (**double** Tan(**void**));
- метод, возвращающий координаты середины отрезка (Segment Center(**void**));
- метод, возвращающий отрезок, перпендикулярный данному и проходящий через точку (x,y) вне прямой, содержащей данный отрезок (Segment Orthogonal(**double** x, **double** y));
- метод, определяющий лежит ли точка на отрезке (**int** Point(**double** x, **double** y));
- метод, который совершает поворот отрезка относительно точки с координатами (x,y) на угол α (Segment Turn (**double** x,**double** y,**double** alfa);
- метод, который возвращает строку с уравнением прямой, на которой лежит отрезок (**string** Line(**void**)).

Операторы:

- < (возвращает 1, если длина отрезка слева меньше длины отрезка справа, 0- в остальных случаях);
- > (возвращает 1, если длина отрезка слева больше длины отрезка справа, 0- в остальных случаях);
- = (возвращает 1, если длины соответствующих отрезков равны);
- * («растягивает» отрезок в N раз, относительно середины отрезка);
- || (совершает параллельный перенос квадрата на вектор (x,y))
- || (возвращает 1, если отрезки лежат на параллельных прямых);
- | (возвращает 1, если отрезки лежат на перпендикулярных прямых);
- + (возвращает сумму длин соответствующих отрезков);
- - (возвращает модуль разности длин соответствующих отрезков).

Вариант 5.

Объект: **Rectangle** (прямоугольник, стороны которого параллельны осям координат):
Данные: 2 точки с координатами концов диагонали (**double** x,y).
Методы:

- длина диагонали (**double** Length(**void**));
- координаты вершин прямоугольника (**double** [4][2] Apex (**void**));
- длины сторон прямоугольника (**double**[2] Leg(**void**));
- метод, возвращающий тангенс угла наклона заданной диагонали к положительному направлению оси OX (**double** Tan(**void**));
- метод, возвращающий координаты пересечения диагоналей (**double**[2] Center(**void**));
- периметр прямоугольника (**double** Perimeter(**void**));
- площадь прямоугольника (**double** Area(**void**));
- метод, который совершает поворот прямоугольника относительно точки с координатами (x,y) на угол α (Rectangle Turn(**double** x, **double** y,**double** alfa);
- метод, который возвращает строку с уравнением прямой, на которой лежит данная диагональ (**string** Line(**void**)).

Операторы:

- < (возвращает 1, если площадь прямоугольника слева меньше площади прямоугольника справа, 0- в остальных случаях),
- > (возвращает 1, если площадь прямоугольника слева больше площади прямоугольника справа, 0- в остальных случаях),
- = (возвращает 1, если площади соответствующих прямоугольников равны);
- * («растягивает» прямоугольник в N раз, относительно центра);
- || (совершает параллельный перенос прямоугольника на вектор (x,y))
- + (возвращает сумму площадей соответствующих прямоугольников);
- - (возвращает модуль разности площадей соответствующих прямоугольников).

Вариант 6.

Объект: **Fraction** (обыкновенная дробь):
Данные: Числитель и знаменатель (**int** denominator, numerator).
Методы:

- метод, определяющий является ли дробь правильной (**int** ProperFraction(**void**));
- метод, определяющий НОД числителя и знаменателя (**int** Divisor(**void**));
- метод, возвращающий сокращенную дробь, равную данной (Fraction DivFraction(**void**));
- метод, возвращающий бесконечную периодическую дробь (с указанием периода), соответствующую данной обыкновенной дроби (**void** Decimal(**void**));
- метод, который возвращает строку с указанием целой и дробной части данной дроби в случае, если дробь неправильная (**string** Implicit(**void**)).

Операторы:

- < (возвращает 1, если дробь слева меньше дроби справа, 0 - в остальных случаях);
- > (возвращает 1, если дробь слева больше дроби справа, 0 - в остальных случаях);
- = (возвращает 1, если дроби равны);
- * (возвращает произведение дробей);
- / (возвращает частное дробей);
- + (возвращает сумму дробей);
- - (возвращает разность дробей).

Вариант 7.

Объект: **Polynom** (Многочлен):

Данные: Степень многочлена, массив коэффициентов при соответствующих степенях, символ, отвечающий за переменную (**int** degree, *coefficient; **char** variable).

Методы:

- метод, определяющий коэффициент многочлена при n-ной степени (**int** Coef(**int** n));
- метод, возвращающий «хвост» многочлена: члены со степенью ниже заданной (Polynom Tail(**int** n));
- метод, возвращающий «голову» многочлена: члены со степенью выше заданной (Polynom Head(**int** n));
- метод, меняющий местами коэффициенты при степенях i, j (Polynom Change (**int** i, **int** j));
- конструктор, инициализирующий объект по строке вида *“a_0*x^0+a_1*x^1+…+a_n*x^n”* Polynom(**char*** line).

Операторы:

- < (возвращает 1, если степень многочлена слева меньше степени многочлена справа, 0- в остальных случаях);
- > (возвращает 1, если степень многочлена слева больше степени многочлена справа, 0- в остальных случаях);
- = (возвращает 1, если многочлены равны);
- * (возвращает произведение многочленов);
- * (возвращает произведение многочлена на число);
- + (возвращает сумму многочленов);
- - (возвращает разность многочленов).

Вариант 8.

Объект: **Vector** (Вектор в N-мерном пространстве):

Данные: Размерность пространства, массив координат вектора (**int** dim, **double*** coord).

Методы:

- длина вектора (**double** Length(**void**));
- косинус угла между двумя векторами (**double** Cosinus(Vector b));
- метод, определяющий являются ли векторы перпендикулярными (**int** Ortos(Vector b));
- метод, определяющий являются ли векторы параллельными (**int** Parallel(Vector b)).

Операторы:

- < (возвращает 1, если длина вектора слева меньше длины вектора справа, 0- в остальных случаях);
- > (возвращает 1, если длина вектора слева больше длины вектора справа, 0- в остальных случаях);
- = (возвращает 1, если длины векторов равны);
- * (возвращает скалярное произведение векторов);
- * (возвращает произведение вектора на число);
- + (возвращает сумму векторов);
- - (возвращает разность векторов).

Вариант 9.

Объект: **Matrix** (Квадратная матрица):

Данные: Степень матрицы, массив элементов матрицы (**int** degree, **double** **element).

Методы:

- определитель матрицы (**double** det(**void**));
- транспонированная матрица (Matrix Transpose(**void**));
- обратная матрица (Matrix Inverse(**void**));
- метод, возвращающий решение матричного уравнения AX=B (X-вектор переменных, B – вектор свободных членов) (**double** * MatrixEq(**double*** B));
- метод, меняющий в матрице местами i-тую строку и j-ый столбец (Matrix Change(**int** i, **int** j));
- конструктор, инициализирующий объект по строке вида "$a_{11}\ a_{12}\ \ldots a_{1n}\backslash n a_{21} \ldots a_{2n}\backslash n \ldots$" Matrix (**char*** a).

Операторы:

- < (возвращает 1, если определитель матрицы слева меньше определителя матрицы справа, 0- в остальных случаях);
- > (возвращает 1, если определитель матрицы слева больше определителя матрицы справа, 0- в остальных случаях);
- = (возвращает 1, если матрицы равны);
- * (возвращает произведение матриц);
- * (возвращает произведение матрицы на число);
- + (возвращает сумму матриц);
- - (возвращает разность матриц).