Контрольные вопросы:

- ❏ (5.б) Перечислите и прокомментируйте основные варианты отношений между классами.
- ❏ (5 б.) Какие существуют разновидности наследования и для чего они предназначены?
- ❏ (5 б.) Что необходимо для корректного функционирования механизма виртуальных функций?
- ❏ (5 б.) Какую проблему решают виртуальные базовые классы при проектировании наследования?
- ❏ (5 б.) В чём заключается принцип подстановки Лисков, и как он помогает при проектировании иерархии классов?

Упражнения:

- ❏ (100 б.) Реализуйте иерархию классов, описывающих геометрические фигуры. За основу можно взять схему из приложения. Ветвь Open Figure реализовывать не нужно, поэтому класс Closed Figure также можно будет исключить.

  Обязательно реализуйте классы, описывающие эллипс, окружность, прямоугольник, квадрат, треугольник. Опционально можете добавить классы параллелограмма и ромба. Реализуйте во всех классах необходимые конструкторы.

  Подумайте над представлением фигур и данными-членами. Следует использовать наиболее общий вариант. Для всех перечисленных подходит вариант с хранением координат x, y заданного количества точек 2D плоскости. В некоторых случаях можно сделать оптимизацию, например, для окружности. Для представления точек плоскости можете создать структуру Point с двумя полями x,y, либо посмотреть библиотечный класс std::pair. Массивы с точками не нужно располагать в каждом классе, достаточно иметь один массив в базовом классе Shape.

  Создайте 3 ветки виртуальных функций: 1) вычисление площади, 2) вычисление периметра и 3) вывод на экран информации о фигуре. В последнем случае должен вызываться оператор вывода, использующий виртуальную функцию print(). Убедитесь, что вы рассмотрели все необходимые аспекты проектирования иерархии классов, которые мы затрагивали на семинаре.

  В main создайте массив фигур, используя указатели на базовый класс. Добавьте в массив разные фигуры и продемонстрируйте работу виртуальных функций-членов.

  Указание: В следующем семестре мы будем работать с мультимедийной библиотекой SFML. Можете взглянуть на её иерархию классов, описывающих геометрические фигуры.

  Примечание: иногда между математическим понятием “является разновидностью” и программным открытым наследованием может возникнуть конфликт. Например, с математической точки зрения, квадрат определённо является разновидностью прямоугольника. Однако в иерархии класс квадрата может унаследовать некоторые функции-члены класса прямоугольника, которые будут с ним несовместимы, например, set_lenght().