Контрольные вопросы:

- ☐ (5 б.) На чем основано объектно-ориентированное программирование?
- ☐ (5 б.) Какие аспекты следует учитывать при проектировании классов?
- ☐ (5 б.) Почему удобно разделять классы на интерефейс и реализацию?
- ☐ (5 б.) Чем внутреннее связывание отличается от внешнего связывания?
- ☐ (5 б.) Какими особенностями обладают именованные пространства имен?

Упражнения:

- ☐ (25 б.) Замените структуру из первой задачи прошлого задания на класс. Реализуйте ее объявление в hpp файле, а определение в cpp файле. Постарайтесь учесть все аспекты проектирования классов, которые мы рассматривали на семинаре – публичный интерфейс, приватные данные, дружественные операторы и проч.
- ☐ (25 б.) (автор – Никита Попов) Разработайте класс, представляющий некоторую физическую величину и предоставляющий интерфейс для записи и чтения её значения в различных единицах измерения. Например, Вы можете разработать класс для энергии, хранить ее значение в джоулях, и реализовать функции-члены для записи и чтения значения с соответствующими преобразованиями в эргах и электрон-вольтах. Для хранения констант, используемых функциями-членами, рекомендуется использовать статические константы.
- ☐ (25 б.) Использование модификатора доступа `friend` делает все компоненты класса доступными для дружественной сущности, даже если ей нужны только некоторые из них, что приводит к ухудшению инкапсуляции и поэтому обычно считается сомнительной практикой. Непосредственная возможность дать доступ только к одной части членов класса для одних пользователей, к другой - для других, а к третьей - третьим отсутствует. Придумайте и реализуйте паттерн, обеспечивающий ограниченный доступ для друзей класса.
- ☐ (25 б.) Смоделируйте ситуации, когда возникает ошибка линковщика с обнаружением многократно определенного внешнего символа и ошибка линковщика с неразрешенным внешним символом. Примечание: здесь Вам не нужно задумываться о внутреннем связывании, рассматривайте только случай внешнего связывания. Как я упоминал на семинаре, внутреннее связывание используется в основном для создания сущностей, доступных только в одном юните трансляции. В наших задачах мы будем работать только с внешним связыванием. Также опишите в комментариях в коде, почему возникают ошибки и как следует их исправить.