1. **На чём основано ООП?**Среди основополагающий концепции ООП можно выделить:

● инкапсуляция - разделение свойств и методов (атрибутов) на те свойства и методы, которые могут быть использованы только внутри класса (приватные/private) и которые могут быть использованы как внутри класса, так и снаружи (публичные/public)

● полиморфизм - разные реализации кода при общем интерфейсе. Пример: перегрузка функций. В С++ реализован Ad-hoc-полиморфизм

(специализированный полиморфизм)

● наследование - передача общих свойств и методов от родителя к потомкам

1. **Какие аспекты следует учитывать при проектировании классов?**  
   Принято располагать private-данные снизу;   
   private-данные начинаются с m\_ или с \_(для понимания, что данные закрытые);   
   конструкторы\деструкторы - в public;

геттеры и сеттеры (инструменты, которые позволяют получить и задать private- данные вне класса) в основном в public.

1. **Почему удобно разделять классы на интерфейс и реализацию?**  
   Интерфейс – совокупность всех публичных методов и переменных, через взаимодействие с интерфейсом осуществляется работа с объектом класса.   
   Реализация – внутреннее устройство класса, она не видна пользователю, но именно благодаря реализации объект выполняет свои задачи. К реализации относится совокупность всех публичных и приватных методов и переменных класса, а также доставшихся ему от родительских классов.  
   Пользователю достаточно работать с интерфейсом, но знание реализации позволяет эффективнее использовать объект.
2. **Чем внутреннее связывание отличается от внешнего связывания?**В единицу трансляции включены файл реализации (.c/.cpp) и все его заголовочные файлы (.h/.hpp). Если внутри единицы трансляции у объекта или функции есть внутреннее связывание, то этот символ виден компоновщику только внутри этой единицы трансляции. Если же у объекта или функции есть внешнее связывание, то компоновщик сможет видеть его при обработке других единиц трансляции.
3. **Какими особенностями обладают именованные пространства имён?**  
   Пространство имен определяет область кода, в которой гарантируется уникальность всех идентификаторов. По умолчанию, глобальные переменные и обычные функции определены в глобальном пространстве имен. Чтобы избежать подобных ситуаций, когда два независимых объекта имеют идентификаторы, которые могут конфликтовать друг с другом при совместном использовании, язык C++ позволяет объявлять собственные пространства имен через ключевое слово namespace. Всё, что объявлено внутри пользовательского пространства имен, — принадлежит только этому пространству имен (а не глобальному).