**Контрольные вопросы:**

**1. На чем основано объектно-ориентированное программирование?**

ООП — метод программирование, основанный на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, где классы образуют иерархию наследования

Основные принципы ООП:

1.Инкапсуляция - принцип, определяющий то как пользователь должен взаимодействовать с классом. Класс должен представлять собой чёрную коробку, пользователь не обязан знать

что находится внутри класса, чтобы в полной мере использовать его функционал. Так же эта концепция говорит о том что класс должен подразделяться на интерфейс - ту часть класса с которой пользователь может непосредственно взаимодействовать, и на приватные данные класса, пользоваться которыми в общем случае может он сам.

2.Наследование - принцип, позволяющий описать новый класс на основе уже существующего.

3.Полиморфизм - принцип, определяющий возможность объектов с одинаковой спецификацией иметь различную реализацию. "Один интерфейс, множество реализаций". Пример полиморфизма обычная перегрузка функций.

4.Абстракция данных. Идея абстракции данных состоит в том, что способ использования составных объектов данных должен отделяться от деталей их реализации в виде более простых объектов.

**2. Важные аспекты при проектировании класса.**

Синтаксическое объявление класса схоже с объявлением структуры. Также в классах реализуются методы (функции-члены), конструкторы, деструкторы.

По умолчанию все данные-члены класса — private, но класс может содержать как закрытые, так и открытые члены.

У классов могут существовать дружественные функции, которые имеют доступ к закрытым членам класса. Также классы можно наследовать.

Про инвариант:

Во время создания классов устанавливаются их инварианты, которые постоянно поддерживаются между вызовами публичных методов. Временное нарушение классовой инвариантности между частными вызовами метода возможно, хотя и нежелательно.

Инвариант объекта представляет собой конструкцию программирования, состоящую из набора свойств. Это гарантирует, что объект всегда будет соответствовать предопределенным условиям, и поэтому методы могут всегда ссылаться на объект без риска сделать неточные презумпции.

**3. Разделение интерфейса и реализации класса.**

Разделение класса на реализацию и интерфейс позволяет менять реализацию/интерфейс, сохраняя интерфейс/реализацию.

Пользователю библиотеки включающей этот класс не нужно будет знать, что находится внутри класса. Класс, его поля и методы представляют собой концепцию "чёрной коробки", в которой находится реализация сокрытая от глаз пользователя

и в которую посредством интерфейса происходит загрузка/выгрузка данных.

**4. Внутреннее и внешнее связывание.**

**Внутренняя связь:**

Если у символа есть внутренняя связь, то он будет виден только внутри текущей единицы трансляции,те компоновщик может использовать этот символ только при обработке единицы трансляции, в которой был объявлен символ. Компилятор будет буквально выделять совершенно новую, уникальную копию для каждой единицы трансляции, где будет использован этот элемент, что может привести к высоким расходам памяти.  
  
Для объявления символа с внутренней связью, существует ключевое слово static.

**Внешняя связь**  
  
Когда символ (переменная или функция) обладает внешней связью, он становится видимым компоновщикам из других файлов, то есть “глобально” видимым, доступным всем единицами трансляции.   
  
Ключевое слово extern объявляет, что у символа есть внешняя связь.

**5. Особенности именованных пространств имен.**

Один и тот же идентификатор(какое-то имя объекта) может быть независимо определён в нескольких пространствах. Таким образом, значение, связанное с идентификатором, определённым в одном пространстве имён, может иметь (или не иметь) такое же значение, как и такой же идентификатор, определённый в другом пространстве. Языки с поддержкой пространств имён определяют правила, указывающие, к какому пространству имён принадлежит идентификатор (то есть его определение).

Вложенность и аддитивность

Вложенное пространство имен является вложенной областью видимости внутри пространства, содержащего его. В процессе разрешения имен вложенные пространства ведут себя так же, как вложенные блоки. Когда некоторое имя употребляется в пространстве имен, поиск его объявление проводится во всех объемлющих пространствах.

Если в пространстве имен A объявлено пространство имен B, то к элементу этого пространства обращаемся A::B::.

Одно пространство имен можно объявить несколько раз, что позволяет распределить его по нескольким файлам и разделить его в одном файле. При компиляции части пространства имен будут соединены в одно целое.