1) О каких принципах следует помнить при разработке функций?

S: Single Responsibility Principle (Принцип единственной ответственности).

O: Open-Closed Principle (Принцип открытости-закрытости).

L: Liskov Substitution Principle (Принцип подстановки Барбары Лисков).

I: Interface Segregation Principle (Принцип разделения интерфейса).

D: Dependency Inversion Principle (Принцип инверсии зависимостей).

2) В чем заключается концепция встраивания вызовов функций?

Одним из главных недостатков использования функций является то, что при каждом их вызове происходит расход ресурсов, что влияет на производительность программы.

C++ предлагает возможность совместить все преимущества функций вместе с высокой производительностью кода, написанного «на месте». Речь идет о встроенных функциях. Ключевое слово inline используется для запроса, чтобы компилятор рассматривал вашу функцию как встроенную.

При компиляции вашего кода, все встроенные функции (англ. «inline functions») раскрываются «на месте», то есть вызов функции заменяется копией содержимого самой функции, и ресурсы, которые могли бы быть потрачены на вызов этой функции, сохраняются.

3) Какие аргументы функции могут иметь значения по умолчанию?

Параметр по умолчанию (или «необязательный параметр») — это параметр функции, который имеет определенное (по умолчанию) значение. Если пользователь не передает в функцию значение для параметра, то используется значение по умолчанию. Если же пользователь передает значение, то это значение используется вместо значения по умолчанию.

**Правило №1: Все параметры по умолчанию в прототипе или в определении функции должны находиться справа.**

**Правило №2: Если имеется более одного параметра по умолчанию, то самым левым параметром по умолчанию должен быть тот, который с наибольшей вероятностью (среди всех остальных параметров) будет явно переопределен пользователем.**

4) На основании чего разрешается выбор перегруженной функции?

Компилятор может определить сам, какую версию функции следует вызывать на основе аргументов, используемых в её вызове. Если параметрами будут переменные типа int, то C++ понимает, что мы хотим вызвать, например f(int, int). Если же мы предоставим два значения типа с плавающей запятой, то C++ поймет, что мы хотим вызвать f(double, double). Фактически, мы можем определить столько перегруженных функций f(), сколько хотим, до тех пор, пока каждая из них будет иметь свои (уникальные) параметры.

5) Как обеспечить «состояние» в функциях и лямбда-выражениях?

Любой идентификатор (переменная, функция) может иметь внешнее или внутреннее связывание,  
но тогда, две функции с одинаковыми именами не могут существовать в пределах одного проекта или всех единиц трансляции.  
Существует ключевое слово static, которое говорит, что функция или переменная не  
экспортируется из модуля, а используется внутри него (внутреннее связывание).  
  
extern указывает на внешнее связывание, но обычно является по умолчанию, но static переменной глобальной и внутри тела функции – это разные вещи. Последнее указывает статическую переменную сохраняющие значение между вызовами функций.