1. Идеальная передача реализуется с помощью std::forward. Она изменяется тип переменной на rvalue для перемещение, если это необходимо и не совершает этого, если не необходимо. Также, если я правильно понимаю, к идеальной передаче можно отнести прием аргументов по универсальной ссылке void function (int && arg) {}. Она может корректно принять и lvalue, и rvalue, и поддерживает добавление константности.
2. Перебрасывающая ссылка это просто T&, с ней производится низведение типа. При переходе к универсальным ссылкам определяется не только тип переданного в функцию параметра, но также даётся оценка, является ли он *rvalue* или *lvalue*.
3. Аббревиатура SFINAE расшифровывается как *substitution failure is not an error.* При определении перегрузок функции ошибочное инстанцирование шаблонов не вызывают ошибку компиляции, а отбрасываются из списка кандидатов на наиболее подходящую перегрузку. То есть при ошибке инстанцирования шаблона компилятор начинает искать другую существующую реализацию этой функции.
4. Он позволяет показать компилятору, что определенный шаблон должен быть скомпилирован только когда в enable_if условие true, а в любом другом случае он должен искать другую реализацию данной функции. И по идиоме SFINAE компилятор не выбрасывает ошибку из-за неправильного инстанцирования шаблона
5. Тип может быть выведен из аргументов неявно компилятором. Тип может быть указан явно. Для типа функции возвращаемого значения можем использовать auto или common_type.
   Также производится низведение типов аргументов, то есть выполняются првила свёртывания ссылок
   - template<typename T>
     void f (T & arg) {}
     int x = 42; f(x) -> T = int; arg = int&
     const int cx = x; f(cx) -> T = const int; arg = const int&
     const int &rx = x; f(rx) -> T = const int; arg = const int&
   - template <typename T>
     void f (T && arg) {} // && - универсальная ссылка

     int x = 42; f(x) -> T = int&; arg = int&
     const int cx = x; f(cx) -> T = const int&; arg = const int&
     const int &rx = x; f(rx) -> T = const int&; arg = const int&
     42 f(42) - > T = int; arg = int&&

   - template <typename T>
     void f (T arg) {}

     Низведения типов не происходит