# Тема 7: Обратное проектирование алгоритма

## 📘 Определение

**Обратное проектирование алгоритма (Reverse Engineering)** — это **процесс анализа алгоритма или программы**, при котором необходимо восстановить:

* исходную **структуру**,
* **логику выполнения**,
* **назначение** алгоритма,

исходя **только из его поведения, выходных данных или скомпилированной версии**, при этом **не имея доступа к оригинальному исходному коду**.

Это процесс «обратного хода»: от результата — к начальной идее.

## 🎯 Цель обратного проектирования

* **Понять**, как работает неизвестный алгоритм или программа.
* **Восстановить** потерянный или устаревший алгоритм.
* **Анализировать** поведение программы, если нет документации.
* **Создать аналог** уже существующего алгоритма (например, при миграции на другую платформу).
* **Оценить эффективность** и найти слабые места (например, в безопасности).
* **Подготовиться к олимпиадам или собеседованиям**, где часто нужно понять суть алгоритма по результатам его выполнения.

## 📦 Основные этапы обратного проектирования

### 1. **Сбор информации**

* Входные и выходные данные
* Логи
* Поведение при разных условиях
* Ошибки, задержки и ответы

### 2. **Формирование гипотез**

* Что делает алгоритм?
* Какие структуры данных могут использоваться?
* Есть ли известные паттерны (например, сортировка, поиск)?

### 3. **Анализ повторяющихся закономерностей**

* Сравнение поведения на разных входах
* Построение таблиц вход–выход
* Нахождение инвариантов (что остаётся неизменным?)

### 4. **Построение модели**

* Сначала — псевдокод, блок-схемы, логические правила
* Затем — возможная реализация на языке программирования

### 5. **Тестирование**

* Сравнение результата своего алгоритма с реальным
* Анализ ошибок и корректировка модели

## Заключение

Обратное проектирование алгоритма — это навык **анализировать, восстанавливать и воссоздавать** алгоритмы и логику работы программ. Он широко используется в профессиональной разработке, кибербезопасности и олимпиадной практике. Этот процесс помогает **глубже понять**, как устроен код, даже если ты видишь только его результат.