**Гибкая методология разработки** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Agile software development*, *agile-методы*) — серия подходов к [разработке программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), ориентированных на использование [итеративной разработки](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0), динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.

Существует несколько методик, относящихся к классу гибких методологий разработки, в частности [экстремальное программирование](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [DSDM](https://ru.wikipedia.org/wiki/DSDM), [Scrum](https://ru.wikipedia.org/wiki/Scrum), [FDD](https://ru.wikipedia.org/wiki/Feature_driven_development).

Применяется как эффективная практика организации труда небольших групп (которые делают однородную творческую работу) в объединении с их управлением комбинированым (либеральным и демократическим) методом.

Большинство гибких методологий нацелены на минимизацию рисков путём сведения разработки к серии коротких циклов, называемых итерациями, которые обычно длятся две-три недели. Каждая итерация сама по себе выглядит как программный проект в миниатюре и включает все задачи, необходимые для выдачи мини-прироста по функциональности: планирование, [анализ требований](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9), [проектирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), [программирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [тестирование](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) и [документирование](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Хотя отдельная итерация, как правило, недостаточна для выпуска новой версии продукта, подразумевается, что гибкий [программный проект](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82) готов к выпуску в конце каждой итерации. По окончании каждой итерации команда выполняет переоценку приоритетов разработки.

Agile-методы делают упор на непосредственное общение лицом к лицу. Большинство agile-команд расположены в одном офисе, иногда называемом [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *bullpen*. Как минимум, она включает и «заказчиков» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *product owner* — заказчик или его полномочный представитель, определяющий требования к продукту; эту роль может выполнять менеджер проекта, бизнес-аналитик или клиент). Офис может также включать тестировщиков, дизайнеров интерфейса, технических писателей и менеджеров.

Основной [метрикой](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) agile-методов является рабочий продукт. Отдавая предпочтение непосредственному общению, agile-методы уменьшают объём письменной документации по сравнению с другими методами. Это привело к критике этих методов как недисциплинированных.

**История**

В феврале 2001 в штате Юта США был выпущен «Манифест гибкой методологии разработки программного обеспечения». Он являлся альтернативой управляемым документацией, «тяжеловесным» практикам разработки программного обеспечения, таким как «[метод водопада](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)», являвшимся золотым стандартом разработки в то время. Данный манифест был одобрен и подписан представителями методологий: [экстремального программирования](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [Crystal Clear](https://ru.wikipedia.org/w/index.php?title=Crystal_Clear&action=edit&redlink=1), [DSDM](https://ru.wikipedia.org/wiki/DSDM), [Feature driven development](https://ru.wikipedia.org/wiki/Feature_driven_development), [Scrum](https://ru.wikipedia.org/wiki/Scrum), [Adaptive software development](https://en.wikipedia.org/wiki/Adaptive_software_development), [Pragmatic Programming](https://ru.wikipedia.org/w/index.php?title=Pragmatic_Programming&action=edit&redlink=1). Гибкая методология разработки использовалась многими компаниями и до принятия манифеста, однако именно после этого события произошло вхождение Agile-разработки в массы.

**Принципы**

Agile — семейство процессов разработки, а не единственный подход в разработке программного обеспечения, и определяется Agile Manifesto. Agile не включает практик, а определяет ценности и принципы, которыми руководствуются успешные команды.

Agile Manifesto разработан и принят 11-13 февраля 2001 года на лыжном курорте The Lodge at Snowbird в горах Юты. Манифест подписали представители следующих методологий: [Extreme programming](https://ru.wikipedia.org/wiki/Extreme_programming), [Scrum](https://ru.wikipedia.org/wiki/Scrum), [DSDM](https://ru.wikipedia.org/wiki/DSDM), [Adaptive software development](https://ru.wikipedia.org/w/index.php?title=Adaptive_software_development&action=edit&redlink=1), [Crystal Clear](https://ru.wikipedia.org/w/index.php?title=Crystal_Clear&action=edit&redlink=1), [Feature driven development](https://ru.wikipedia.org/wiki/Feature_driven_development), [Pragmatic Programming](https://ru.wikipedia.org/w/index.php?title=Pragmatic_Programming&action=edit&redlink=1). Agile Manifesto содержит 4 основные идеи и 12 принципов. Примечательно, что [Agile Manifesto](http://www.agilemanifesto.org/iso/ru/) не содержит практических советов.

Основные идеи:

* люди и взаимодействие важнее процессов и инструментов;
* работающий продукт важнее исчерпывающей документации;
* сотрудничество с заказчиком важнее согласования условий контракта;
* готовность к изменениям важнее следования первоначальному плану.

Принципы, которые разъясняет Agile Manifesto:

* удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
* приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
* частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
* тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
* проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
* рекомендуемый метод передачи информации — личный разговор (лицом к лицу);
* работающее программное обеспечение — лучший измеритель прогресса;
* спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
* постоянное внимание улучшению технического мастерства и удобному дизайну;
* простота — искусство не делать лишней работы;
* лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
* постоянная адаптация к изменяющимся обстоятельствам.

**Критика**

Один из повторяющихся пунктов критики: при agile-подходе часто пренебрегают созданием плана («дорожной карты») развития продукта, равно как и [управлением требованиями](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8), в процессе которого и формируется такая «карта». Гибкий подход к управлению требованиями не подразумевает далеко идущих планов (по сути, управления требованиями просто не существует в данной методологии), а подразумевает возможность заказчика вдруг и неожиданно в конце каждой итерации выставлять новые требования, часто противоречащие архитектуре уже созданного и поставляемого продукта. Такое иногда приводит к катастрофическим «авралам» с массовым [рефакторингом](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) и переделками практически на каждой очередной итерации.

Кроме того, считается, что работа в agile мотивирует разработчиков решать все поступившие задачи простейшим и быстрейшим возможным способом, при этом зачастую не обращая внимания на правильность кода с точки зрения требований нижележащей платформы (подход — «работает, и ладно», при этом не учитывается, что может перестать работать при малейшем изменении или же дать тяжёлые к воспроизводству дефекты после реального развёртывания у клиента). Это приводит к снижению качества продукта и накоплению дефектов.

**Методологии**

Существуют методологии, которые придерживаются ценностей и принципов заявленных в Agile Manifesto, некоторые из них:

* [Agile Modeling](https://ru.wikipedia.org/w/index.php?title=Agile_Modeling&action=edit&redlink=1) ([*англ.*](https://en.wikipedia.org/wiki/Agile_Modeling)) — набор понятий, принципов и приёмов (практик), позволяющих быстро и просто выполнять моделирование и документирование в проектах разработки программного обеспечения. Не включает в себя детальную инструкцию по проектированию, не содержит описаний, как строить диаграммы на UML. Основная цель: эффективное моделирование и документирование; но не охватывает программирование и тестирование, не включает вопросы управления проектом, развёртывания и сопровождения системы. Однако включает в себя проверку модели кодом[[3]](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8#cite_note-3).
* [Agile Unified Process](https://ru.wikipedia.org/w/index.php?title=Agile_Unified_Process&action=edit&redlink=1) ([*англ.*](https://en.wikipedia.org/wiki/Agile_Unified_Process)) (AUP) упрощенная версия IBM Rational Unified Process ([RUP](https://ru.wikipedia.org/wiki/RUP)), разработанная Скоттом Амблером, которая описывает простое и понятное приближение (модель) для создания программного обеспечения для бизнес-приложений.
* [Agile Data Method](https://ru.wikipedia.org/w/index.php?title=Agile_Data_Method&action=edit&redlink=1) ([*англ.*](https://en.wikipedia.org/wiki/Agile_Data_Method)) — группа итеративных методов разработки программного обеспечения, в которых требования и решения достигаются в рамках сотрудничества разных кросс-функциональных команд.
* [DSDM](https://ru.wikipedia.org/wiki/DSDM) основан на концепции быстрой разработки приложений (Rapid Application Development, RAD). Представляет собой итеративный и инкрементный подход, который придаёт особое значение продолжительному участию в процессе пользователя/потребителя.
* [Essential Unified Process](https://ru.wikipedia.org/w/index.php?title=Essential_Unified_Process&action=edit&redlink=1) ([*англ.*](https://en.wikipedia.org/wiki/Essential_Unified_Process)) (EssUP).
* [Экстремальное программирование](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B5%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Extreme programming*, XP).
* [Feature driven development](https://ru.wikipedia.org/wiki/Feature_driven_development) (FDD) — функционально-ориентированная разработка. Используемое в FDD понятие функции или свойства ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *feature*) системы достаточно близко к понятию прецедента использования, используемому в RUP, существенное отличие — это дополнительное ограничение: «каждая функция должна допускать реализацию не более, чем за две недели». То есть если сценарий использования достаточно мал, его можно считать функцией. Если же велик, то его надо разбить на несколько относительно независимых функций.
* [Getting Real](https://ru.wikipedia.org/w/index.php?title=Getting_Real&action=edit&redlink=1) — итеративный подход без функциональных спецификаций, использующийся для веб-приложений. В данном методе сперва разрабатывается интерфейс программы, а потом её функциональная часть.
* [OpenUP](https://ru.wikipedia.org/wiki/OpenUP) — это итеративно-инкрементальный метод разработки программного обеспечения. Позиционируется как лёгкий и гибкий вариант [RUP](https://ru.wikipedia.org/wiki/RUP). OpenUP делит жизненный цикл проекта на четыре фазы: начальная фаза, фазы уточнения, конструирования и передачи. Жизненный цикл проекта обеспечивает предоставление заинтересованным лицам и членам коллектива точек ознакомления и принятия решений на протяжении всего проекта. Это позволяет эффективно контролировать ситуацию и вовремя принимать решения о приемлемости результатов. План проекта определяет жизненный цикл, а конечным результатом является окончательное приложение.
* [Scrum](https://ru.wikipedia.org/wiki/Scrum) устанавливает правила управления процессом разработки и позволяет использовать уже существующие практики кодирования, корректируя требования или внося тактические изменения. Использование этой методологии дает возможность выявлять и устранять отклонения от желаемого результата на более ранних этапах разработки программного продукта.
* [Бережливая разработка программного обеспечения](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%B5%D0%B6%D0%BB%D0%B8%D0%B2%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *lean software development*) использует подходы из концепции [бережливого производства](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%B5%D0%B6%D0%BB%D0%B8%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE).

**Бережливая разработка программного обеспечения** — [методология разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), использующая методы концепции [бережливого производства](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%B5%D0%B6%D0%BB%D0%B8%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE). Возникла из среды сторонников концепции [гибкой методологии разработки](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8).

## Принципы

* **Исключение потерь**. Потерями считается всё, что не добавляет ценности для потребителя. В частности: излишняя функциональность; ожидание (паузы) в процессе разработки; нечёткие требования; бюрократизация; медленное внутреннее сообщение.
* **Акцент на обучении.** Короткие циклы разработки, раннее тестирование, частая [обратная связь](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%81%D0%B2%D1%8F%D0%B7%D1%8C) с заказчиком.
* **Предельно отсроченное принятие решений.** Решение следует принимать не на основе предположений и прогнозов, а после открытия существенных фактов.
* **Предельно быстрая доставка заказчику.** Короткие итерации.
* **Мотивация команды.** Нельзя рассматривать людей исключительно как ресурс. Людям нужно нечто большее, чем просто список заданий.
* **Интегрирование.** Передать целостную информацию заказчику. Стремиться к целостной архитектуре. [Рефакторинг](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3).
* **Целостное видение.** Стандартизация, установление отношений между разработчиками. Разделение разработчиками принципов бережливости. «Мыслить широко, делать мало, ошибаться быстро; учиться стремительно».

## Практики

Некоторые практики бережливой разработки аналогичны практикам быстрой разработки, а некоторые несколько различаются. Примеры практик:

* Обнаружение потерь («[en:Muda (Japanese term)](https://en.wikipedia.org/wiki/Muda_%28Japanese_term%29" \o "en:Muda (Japanese term))»)
* [Систематизирование потока ценности](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%B0_%D1%86%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1) (Value stream mapping)
* [Теория ограничений](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D0%B9)
* [«Вытягивающая»](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%82%D1%8F%D0%B3%D0%B8%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE) система ([Канбан](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B1%D0%B0%D0%BD" \o "Канбан))
* [Теория массового обслуживания](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%BC%D0%B0%D1%81%D1%81%D0%BE%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D1%81%D0%BB%D1%83%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)
* [Мотивация](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%82%D0%B8%D0%B2%D0%B0%D1%86%D0%B8%D1%8F)
* Измерения