SOFTWARE PERFORMANCE TESTING

Shauchuk V.A. (Russian Federation)

Shauchuk Volha Andreyevna - Quality Assurance Engineer,
INTERNATIONAL SOFTWARE COMPANY 'EPAM SYSTEMS,
VITEBSK

Abstract: *the article analyses the key aspects and methodologies of software performance testing in the modern information world. The main goals of this article are: explaining the importance of performance testing to ensure high-level software operation in dynamic IT market conditions; enlightening about the methods and tools used to evaluate software performance and ensure its high-level functioning; providing recommendations and methods to enhance application performance aiming at improving their efficiency and meeting user needs; exploring trends and development prospects in the field of performance testing, assessing the impact of load factors on performance.*
Keywords: *software, information technology, performance testing, metrics and testing methods, practical application, testing tools, trend analysis and prospects, user experience.*

ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Шевчук О.А. (Российская Федерация)

Шевчук Ольга Андреевна - инженер по контролю качества программного обеспечения,
Международная компания программного обеспечения «EPAM Systems»,
г. Витебск

Аннотация: *в статье анализируются ключевые аспекты и методики тестирования производительности программного обеспечения в современном информационном мире. Основными целями данной статьи являются: объяснение важности тестирования производительности для обеспечения высокого уровня работы программного обеспечения в динамичных условиях рынка информационных технологий; просвещение о методах и инструментах, которые используются для оценки производительности ПО и обеспечения высокого уровня его функционирования; предоставление рекомендаций и методов улучшения производительности приложений с целью повышения их эффективности и удовлетворения потребностей пользователей, исследование тенденций и перспектив развития в области тестирования производительности, оценка влияния факторов нагрузки на производительность.*
Ключевые слова: *программное обеспечение, информационные технологии, тестирование производительности, метрики и методы тестирования, практическое применение, инструменты тестирования, исследование тенденций и перспектив, пользовательский опыт.*

Введение

"Совершенство основывается не на том, чтобы много делать, а на том, чтобы то, что делаешь, делать хорошо" - Ян Будяшек.

Тестирование программного обеспечения — это неотъемлемая часть разработки программного обеспечения, которая направлена на обеспечение качества, надежности и эффективности программных продуктов. Тестирование программного обеспечения включает в себя проверку и валидацию функциональности, производительности, безопасности, удобства использования и т. д. программных продуктов по отношению к требованиям и спецификациям [1]. В эпохе цифрового прогресса, где виртуальное становится реальностью, и скорость работы приложений имеет значение не менее, чем их функциональность, тестирование производительности программного обеспечения становится определяющим фактором успеха. Все больше пользователей ожидают от приложений не только широкий функционал, но и непрерывную, мгновенную и безукоризненную работу в любых условиях и под любыми нагрузками. Тестирование производительности перестало быть дополнительным этапом разработки и превратилось в неотъемлемую часть цикла создания программного обеспечения. Оно стало ключевым фактором обеспечения конкурентоспособности и удовлетворения требований современных пользователей, для которых быстрая, надежная и эффективная работа приложений — это не просто преимущество, но и необходимость, а также это искусство баланса между ожиданиями пользователей, требованиями бизнеса и возможностями технологии. Тестирование производительности (англ. Performance Testing) в инженерии программного обеспечения — тестирование, которое проводится с целью определения, как быстро работает вычислительная система или её часть под определённой нагрузкой [2]. Основная цель тестирования производительности - выявить и устранить узкие места

производительности в программном приложении [3]. Многие тесты на производительность делаются без попытки осмыслить их реальные цели. Перед началом тестирования всегда должен быть задан бизнес-вопрос: «Какую цель мы преследуем, тестируя производительность?». Ответы на этот вопрос являются частью технико-экономического обоснования (или business case) тестирования. Цели могут различаться в зависимости от технологий, используемых приложением, или его назначения [2].

Методы тестирования производительности ПО

Нагрузочное тестирование - проверяет, может ли программное обеспечение оптимально работать при ожидаемом количестве пользователей. Цель состоит в том, чтобы выявить узкие места в производительности и устранить их до запуска приложения.

Стресс-тестирование - проверяет максимальный предел прочности приложения. Это предполагает использование экстремальных рабочих нагрузок и высокого трафика для тестирования при сбое приложения [4].

Тестирование стабильности — это специализированный процесс, направленный на проверку устойчивости программного обеспечения (ПО) при различных условиях эксплуатации. Основное внимание уделяется выявлению и устранению проблем, которые могут привести к сбоям, зависаниям или другим негативным эффектам в работе приложения [5].

Конфигурационное тестирование — специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.) [6].

Инструменты и метрики, используемые при тестировании производительности, играют ключевую роль в оценке работы программного обеспечения.

Apache JMeter — это инструмент нагрузочного тестирования на основе протокола. JMeter имитирует трафик и одновременных пользователей. Он похож на Locust, но работает быстрее. Apache ab — это инструмент для бенчмаркинга HTTP-серверов. Как таковой, он охватывает гораздо меньшую часть вашей системы, чем Apache JMeter — только HTTP-серве [7]. LoadRunner - коммерческое приложение, разработанное для проведения тестирования производительности, проверки устойчивости системы под нагрузкой, а также для анализа производительности и масштабируемости. Gatling - инструмент для тестирования производительности и нагрузочного тестирования веб-приложений. Gatling позволяет создавать сценарии тестирования на языке программирования Scala [8].

Метрики тестирования программного обеспечения: Время отклика (TTFB) — время, затраченное на получение первого байта ответа от сервера. Время загрузки страницы (PLT) — время, затраченное на загрузку всей страницы, включая все ресурсы. Количество запросов (REQ) — количество HTTP-запросов, необходимых для загрузки страницы. Количество ошибок (ERR) — количество HTTP-ошибок, произошедших во время загрузки страницы. Количество одновременных пользователей (VUs) — количество пользователей, работающих с системой одновременно. Центральный процессор (CPU) — использование процессора системой в процентах. Среднее время между отказами (MTBF) — среднее время, прошедшее между сбоями системы. Среднее время восстановления (MTTR) — среднее время, затраченное на восстановление системы после сбоя. [9]

Проблемы с производительностью

Длительное время загрузки - обычно время загрузки — это начальное время, необходимое приложению для запуска. Обычно оно должно быть сведено к минимуму;

Плохое время отклика. Время отклика — это время, которое проходит с момента ввода пользователем данных в приложение до того, как приложение выдаст ответ на этот ввод;

Плохая масштабируемость - программный продукт страдает от плохой масштабируемости, когда он не может обрабатывать ожидаемое количество пользователей.

Узкие места (Bottlenecking) — это препятствия в системе, которые ухудшают общую производительность системы. Узкое место — это либо ошибки в коде, либо проблемы с оборудованием, которые вызывают снижение пропускной способности при определенных нагрузках. Узкие места обычно устраняются путем исправления плохо работающих процессов или добавления дополнительного оборудования. Некоторые общие узкие места производительности: Загрузка ЦП; Использование памяти; Использование сети; Ограничения операционной системы; Использование диска [3].

Масштабируемость приложений

Эффективная масштабируемость приложений позволяет им расти вместе с потребностями бизнеса и пользователей, обеспечивая высокую производительность и доступность даже при увеличении нагрузки или объема данных. Горизонтальное масштабирование — увеличение степени параллелизма, добавление новых серверов, выполняющих одни и те же функции: серверов БД, серверов приложений, балансировщиков и т. д. В данном случае применяется типовая конфигурация компонентов системы, но увеличивается их общее количество. Вертикальное масштабирование —

увеличение производительности отдельного компонента системы для повышения производительности всей системы в целом: добавление оперативной памяти на сервере, замена процессора на более производительный, добавление более скоростного накопителя и так далее. В данном случае архитектура системы остаётся точно такой же, но улучшается определённое звено [10].

Процесс тестирования производительности

Определите свою среду тестирования: знайте свою физическую тестовую среду, производственную среду и доступные инструменты тестирования. Определите критерии приемлемости производительности: сюда входят цели и ограничения по пропускной способности, времени отклика и распределению ресурсов. Планирование и разработка тестов производительности: определите, как использование может различаться среди конечных пользователей, и определите ключевые сценарии для тестирования всех возможных вариантов использования. Настройка тестовой среды, реализуйте дизайн теста, запустите тесты; Анализируйте, настраивайте и повторно тестируйте. Объединяйте, оценивайте и сообщайте результаты тестов путем анализа, настройки и повторного тестирования [11].

Тестирование производительности примеры

Когда 1000 пользователей одновременно заходят на сайт, убедитесь, что время ответа составляет менее 4 секунд. Если сетевое соединение медленное, убедитесь, что время ответа приложения под нагрузкой находится в допустимом диапазоне. Прежде, чем приложение выйдет из строя, проверьте максимальное количество пользователей, которыми оно может управлять. Когда одновременно читаются/записываются 500 записей, проверьте время выполнения базы данных. Проверьте загрузку процессора и памяти приложения и сервера базы данных во время высоких нагрузок. Проверьте время реакции приложения при низкой, средней, умеренной и высокой нагрузке.

Примеры успешного и неудачного тестирования производительности программного обеспечения

Netflix — это сервис потокового видео с более 200 миллионами пользователей, который тестирует свои приложения с помощью Chaos Monkey, Chaos Engineering, Simian Army и других инструментов, имитирующих сбои и непредвиденные ситуации [12, 13]. **Facebook** — это социальная сеть с более 2,8 миллиардами пользователей в месяц, которая тестирует свои приложения с помощью Quick Performance Tests, Performance Regression Framework, Mobile Lab и других инструментов, измеряющих и оптимизирующих время загрузки, потребление памяти, энергии и другие параметры [14]. **Google** — это компания в области поиска, рекламы, облачных вычислений и искусственного интеллекта с более 4 миллиардами пользователей в день, которая тестирует свои приложения с помощью PageSpeed Insights, Lighthouse, Web Vitals и других инструментов, анализирующих и улучшающих скорость, доступность, интерактивность и другие аспекты производительности [15].

Однако тестирование производительности не всегда успешно и может привести к различным ошибкам и проблемами:

Использование неправильного времени задержки или темпа в тестах. Например, если время пользователя равно 1 секунде, то приложение должно работать быстро и без задержек. Если же время пользователя равно 10 секундам, то приложение должно работать медленно и с задержками.

Использование неправильных параметров или условий для тестирования. Например, если приложение работает на базе данных, то тестер должен использовать данные из базы данных для проверки его производительности. Если же тестер использует данные из другого источника или создает свои данные для тестирования, то он может получить неверные или неточные результаты.

Использование недостаточного количества или качества данных для тестирования. Слишком мало данных может привести к недостаточной проверке различных факторов производительности, таких как количество запросов, количество ответов, количество ошибок и другие. Слишком много данных может привести к перегрузке системы и потере точности результатов [16, 17].

Исследование тенденций и перспектив развития в области тестирования производительности

Внедрение автоматизированного тестирования: автоматизированное тестирование производительности становится неотъемлемой частью разработочного цикла. Это позволяет выявлять проблемы производительности на ранних стадиях и быстро реагировать на них [7]. Пример: использование инструментов типа Selenium Grid для распределенного тестирования веб-приложений. **Тестирование микросервисов и облачных решений:** растущая популярность микросервисной архитектуры и облачных сервисов требует разработки методов тестирования и мониторинга производительности в таких окружениях [18]. Пример: инструменты типа Kubernetes для контейнеризации и управления микросервисами. **Тестирование на реальных устройствах и разнообразных платформах:** с увеличением разнообразия устройств и платформ (мобильные устройства, различные операционные системы) появляется необходимость в тестировании на реальных устройствах [19]. Пример: использование платформы AWS Device Farm для тестирования мобильных приложений на разных устройствах. **Применение искусственного интеллекта и анализ данных:** использование AI и машинного обучения для анализа данных производительности и выявления

аномалий [20]. Пример: использование AI для прогнозирования и оптимизации производительности приложений в реальном времени. **Симуляция реальных условий:** для этого необходимо создать тестовую среду, которая воспроизводит основные параметры и характеристики реальной среды. Например, можно использовать виртуальные машины, сетевое окружение, а также специализированные инструменты для моделирования различных условий работы ПО [19].

Рекомендации по улучшению производительности

Оптимизация базы данных: добавление индексов к таблицам для ускорения выполнения запросов. Пример: после добавления индексов к полям, используемым в частых запросах, время выполнения запросов сократилось на 40%. **Кэширование данных:** использование кэширования для хранения часто запрашиваемых данных и уменьшения нагрузки на сервер. Пример: после внедрения кэширования времени отклика на запросы улучшилось на 25%. **Оптимизация кода:** проверка и оптимизация алгоритмов для снижения времени выполнения операций. Пример: после пересмотра алгоритма сортировки, время сортировки данных сократилось с 10 до 5 секунд. **Использование CDN для статических ресурсов:** для распределения статических ресурсов и ускорения их загрузки для пользователей. Пример: после использования CDN, время загрузки изображений на сайте уменьшилось на 30%. **Оптимизация клиентской части:** оптимизация клиентских скриптов для снижения их размера и улучшения загрузки страниц. Пример: после минификации и сжатия клиентских скриптов, время загрузки страницы уменьшилось на 20%. **Оценка эффективности оптимизаций:** после выявления проблемных зон системы и внесения изменений проводится повторное тестирование для сравнения производительности до и после оптимизации. Анализ результатов позволяет оценить эффективность внесенных изменений и их влияние на производительность, пример: предположим, оптимизировали запросы к базе данных CRM-системы, что сократило время выполнения запросов на 30%. После повторного тестирования обнаружили, что время доступа к информации о клиентах уменьшилось, что является показателем эффективности проведенных оптимизаций.

Выводы

Важность тестирования производительности: эффективное тестирование производительности необходимо для обеспечения стабильной работы приложений в условиях растущих требований пользователей и динамичного рынка технологий.

Комплексный подход к тестированию: успешное тестирование производительности требует комплексного подхода, включающего не только технические метрики, но и аспекты безопасности, управление рисками и удовлетворение пользовательских потребностей.

Интеграция в разработку: непрерывное тестирование производительности включает его интеграцию в процесс разработки, что позволяет выявлять и устранять проблемы на ранних этапах итераций разработки.

Использование метрик: использование четких метрик позволяют быстро и точно оценить производительность, что ускоряет процесс принятия решений по улучшению приложения. Тестирование производительности должно проводиться регулярно, чтобы самый высокопроизводительный сайт или программное обеспечение продолжали выполнять свои ожидаемые функции. Непрерывное тестирование производительности означает, что любые проблемы, которые могут возникнуть в реальном времени, решаются как можно быстрее. Если веб-сайт постоянно удобен для пользователей и совершенствуется, чтобы никогда не отставать, клиенты будут часто его посещать [21].

Заключение

Тестирование производительности программного обеспечения занимает центральное место в создании качественных и конкурентоспособных приложений. Это процесс, который не только измеряет производительность системы, но и вносит существенный вклад в улучшение её работы. В современном мире, где пользователи требуют высокой скорости, стабильности и удобства работы с приложениями, тестирование производительности становится критически важным. Оно позволяет выявить и устранить узкие места, повысить отзывчивость системы, обеспечить её стабильность при различных нагрузках и улучшить общее восприятие пользователем. Основными методами, такими как нагрузочное, стресс-тестирование и анализ производительности, тестировщики оценивают работу системы в реальных условиях и при критических нагрузках. Использование специализированных инструментов позволяет объективно измерить производительность, идентифицировать узкие места и предложить улучшения. Кроме того, тестирование производительности не ограничивается одним этапом разработки. Это непрерывный процесс, который должен сопровождать всю жизненную цикл программного продукта. Постоянное развитие, оптимизация и адаптация к изменяющимся условиям обеспечивают высокий уровень производительности на протяжении всего существования приложения. Как сказал известный американский ученый и писатель Джеральд Вейнберг: «Если тестирование — это процесс поиска ошибок, то ошибки всегда будут найдены» - тестирование

производительности — это не одноразовый процесс, а непрерывная деятельность, которая должна проводиться на всех этапах жизненного цикла приложения, от планирования и проектирования до развертывания и поддержки. Таким образом, тестирование производительности является неотъемлемой частью процесса разработки ПО, гарантируя не только стабильность и надежность приложения, но и удовлетворение потребностей и ожиданий пользователей в быстродействии и комфорте использования.

Список литературы / References

1. *Шевчук В.И.* Современные подходы к тестированию программного обеспечения. [Электронный ресурс]. 2023. URL: https://vc.ru/u/1205966-vital-shauchuk/691440-sovremennye-podhody-k-testirovaniyu-programmnogo-obespecheniya (Дата обращения: 10.10.2023).
2. Wikipedia Тестирование производительности. [Электронный ресурс]. 2015. URL: 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%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8#:~:text=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20(%D0%B0%D0%BD%D0%B3%D0%BB.%20Performance%20Testing),%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C%2C%20%D0%BD%D0%B0%D0%B4%D1%91%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B8%20%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%20%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2 (Дата обращения: 10.12.2023).
3. Библия QA Тестирование производительности (Performance testing). [Электронный ресурс]. URL: https://vladislaveremeev.gitbook.io/qa_bible/vidy-metody-urovni-testirovaniya/testirovanie-proizvoditelnosti-performance-testing (Дата обращения: 08.12.2023).
4. *Терешков Б.* Учебник по тестированию производительности: автоматизация, Gatling и Jenkins. [Электронный ресурс]. 2021. URL: https://nanomode.ru/coding2/uchebnik-po-testirovaniyu-proizvoditelnosti-avtomatizatsiya-gatling-i-jenkins/ (Дата обращения: 18.12.2023).
5. QA evolution Тестирование стабильности. [Электронный ресурс]. 2023. URL: https://qaevolution.ru/testirovanie-stabilnosti/ (Дата обращения: 13.12.2023).
6. QA Bible Конфигурационное тестирование (Configuration testing). [Электронный ресурс]. https://vladislaveremeev.gitbook.io/qa_bible/vidy-metody-urovni-testirovaniya/konfiguracionnoe-testirovanie-configuration-testing (Дата обращения: 13.12.2023).
7. Testengineer.ru Тестирование производительности: теория и немного практики. [Электронный ресурс]. 2021. URL: https://testengineer.ru/testirovanie-proizvoditelnosti-veb-servisov/ (Дата обращения: 13.12.2023).
8. *Рыжов И.* Автоматизированное тестирование производительности: инструменты и практика. [Электронный ресурс]. 2023. URL: https://gochadev.ru/2023/10/10/avtomatizirovannoe-testirovanie-proizvoditelnosti-instrumenti-i-praktika/ (Дата обращения: 10.12.2023).
9. QA evolution Тестирование производительности. [Электронный ресурс]. 2023. URL: https://qaevolution.ru/testirovanie-po/vidy-testirovaniya-po/testirovanie-proizvoditelnosti/ (Дата обращения: 13.12.2023).
10. Dzen.ru Тестирование производительности. Часть 2. [Электронный ресурс]. 2023. URL: https://dzen.ru/a/ZB8y0YxERkhSmRe6 (Дата обращения: 13.12.2023).
11. *Нанда В.* Учебное пособие по тестированию производительности (определение, типы, метрики, пример). [Электронный ресурс]. 2021. URL: https://www.tutorialspoint.com/performance-testing-tutorial-definition-types-metrics-example (Дата обращения: 19.12.2023).
12. Wikipedia Netflix. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Netflix (Дата обращения: 12.12.2023).
13. *Mark Jones* How Netflix pioneered Chaos Engineering. 2019. URL: https://techhq.com/2019/03/how-netflix-pioneered-chaos-engineering/ (Дата обращения: 12.12.2023).
14. Wikipedia Facebook. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Facebook (Дата обращения: 12.12.2023).
15. Wikipedia Google. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Google (Дата обращения: 12.12.2023).
16. LoadView 10 наиболее распространенных ошибок в тестировании производительности. [Электронный ресурс]. 2023. URL: https://www.loadview-testing.com/ru/blog/10-%D0%BD%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D0%B5%D0%B5-

%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%BD%D1%8B%D1%85-%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA-%D0%B2-%D1%82/ (Дата обращения: 09.12.2023).
17. LoadView Типы тестирования программного обеспечения: различия и примеры. [Электронный ресурс]. 2023. URL: https://www.loadview-testing.com/ru/blog/%D1%82%D0%B8%D0%BF%D1%8B-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%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/ (Дата обращения: 09.12.2023).
18. *Chistyakov V.* Секреты разработки высокопроизводительных приложений и микросервисов. [Электронный ресурс]. 2022. URL: https://medium.com/nuances-of-programming/%D1%81%D0%B5%D0%BA%D1%80%D0%B5%D1%82%D1%8B-%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8-%D0%B2%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D1%85-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9-%D0%B8-%D0%BC%D0%B8%D0%BA%D1%80%D0%BE%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D0%BE%D0%B2-41076220ed10 (Дата обращения: 13.12.2023).
19. *Кодов А.* Как проводить тестирование на разных платформах. [Электронный ресурс]. 2023. URL: https://sky.pro/media/kak-provodit-testirovanie-na-raznyh-platformah/ (Дата обращения: 13.12.2023).
20. WSS & Technologies Искусственный интеллект для анализа больших данных (BigData). [Электронный ресурс]. URL: https://www.websoftshop.ru/information/articles/big_data/using_ai_in_data_analysis/ (Дата обращения: 13.12.2023).
21. ZAPTEST Что такое тестирование производительности? Глубокое погружение в типы, практику, инструменты, проблемы и многое другое. [Электронный ресурс]. URL: https://www.zaptest.com/ru/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-%D1%82%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%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB/ (Дата обращения: 19.12.2023).