**Тема:** Цели, задачи и методы исследования программного кода  
**Формат:** Индивидуальный проект  
**Цель:** Глубже осмыслить аспекты, связанные с целями, задачами и методами исследования программного кода

**Практическая часть (список вопросов):**

Цели исследования программного кода

**1. Каковы основные цели анализа программного кода?**

Выявление ошибок и уязвимостей, оценка качества кода, оптимизация производительности, соблюдение стандартов, подготовка к рефакторингу или масштабированию.

**2. Почему исследование программного кода является важным этапом его разработки?**

Позволяет **предотвратить критические ошибки** до релиза.

Улучшает **стабильность и безопасность** ПО.

Снижает **стоимость исправления дефектов** (чем раньше найдена ошибка, тем дешевле её исправить).

**3. Какие результаты могут быть достигнуты благодаря исследованию программного кода?**

Устранение **уязвимостей и багов**, повышение **производительности и эффективности** кода, улучшение **архитектуры и читаемости**, снижение **технического долга**.

**4. Как исследование кода помогает улучшить качество программного обеспечения?**

**Автоматизированный анализ** выявляет слабые места.

**Тестирование** проверяет корректность работы.

**5. Какие долгосрочные цели могут быть связаны с анализом программного кода?**

**Поддержка и масштабируемость** кода в будущем.

**Снижение затрат** на обслуживание.

**Автоматизация проверок** (CI/CD).

**6. Какие задачи решает статический анализ программного кода?**

Поиск **синтаксических ошибок**.

Проверка **стиля кода** и выявление **потенциальных уязвимостей**.

**7. Каковы задачи динамического анализа программного обеспечения?**

Тестирование **в runtime** (производительность, утечки памяти).

Проверка **реакции на входные данные**.

Анализ **поведения приложения** под нагрузкой.

**8. Чем отличаются задачи рефакторинга кода от задач его тестирования?**

| **Рефакторинг** | **Тестирование** |
| --- | --- |
| Улучшение структуры без изменения функционала | Проверка корректности работы |
| Снижение сложности кода | Поиск багов и edge cases |

**9. Какие аспекты документирования кода следует учитывать при исследовании?**

**Комментарии** (но без избыточности).

**README и техническая документация**.

**10. Как провести анализ производительности кода в рамках его исследования?**

**Профилирование**.

**Нагрузочное тестирование**.

**Оптимизация алгоритмов**.

**Методы исследования программного кода**

**11. Какие методы существуют для статического анализа программного кода?**

**Линтеры**.

**Анализаторы безопасности**.

**Метрики кода**.

**12. Какова роль тестирования в исследовании программного обеспечения?**

**Unit-тесты** - проверка отдельных модулей.

**Интеграционные тесты** - проверка взаимодействия компонентов.

**E2E-Тесты**  - имитация действий пользователя.

**13. Какие инструменты используют для выполнения динамического анализа программного кода?**

**Valgrind** (утечки памяти в C/C++).

**Chrome DevTools** (профилирование JS).

**JMeter** (нагрузочное тестирование).

**14. Как метод код-ревью может быть применен в исследовании программного кода?**

**Выявление логических ошибок**.

**Обмен знаниями** внутри команды.

**Соблюдение стиля кода**

**15. Что такое автоматизированные инструменты анализа кода и какие их преимущества?**

**Примеры**: SonarQube, Coverity, Snyk.

**Плюсы**:

Быстрота проверки.

Масштабируемость.

Интеграция в CI/CD.

**16. Какие проблемы могут возникнуть при исследовании программного кода?**

**Ложные срабатывания** (false positives).

**Устаревшая документация**.

**Сложность анализа legacy-кода**.

**17. Как справляться с техническим долгом в процессе анализа кода?**

**Приоритезация** (что критично, а что можно отложить).

**Рефакторинг малыми итерациями**.

**Автоматизация тестов**.

**18. Какие трудности могут возникнуть при интерпретации результатов анализа?**

**Неочевидные метрики** (например, высокая цикломатическая сложность не всегда = плохо).

**Конфликт инструментов** (разные линтеры дают разные рекомендации).

**19. Как обеспечить безопасность при проведении исследований программного кода?**

**Анализ зависимостей** (npm audit, pip-audit).

**SAST-инструменты** (Checkmarx, Semgrep).

**Ручной аудит безопасности**.

**20. В чем заключается важность использования учёта метрик кода?**

**Объективная оценка качества**.

**Выявление проблемных мест**.

**Контроль технического долга**.

**21. Как составить стратегию для исследования программного кода?**

**Определить цели** (безопасность, производительность, читаемость).

**Выбрать инструменты** (линтеры, тесты, метрики).

**Назначить ответственных**.

**Интегрировать в CI/CD**.

**22. Какие метрики кода наиболее полезны при анализе его качества?**

**Цикломатическая сложность**.

**Коэффициент покрытия тестами**.

**Количество строк кода** .

**23. Как проводить анализ кода в условиях Agile-методологий?**

**Встраивать в спринты** (например, SonarQube в PR).

**Автоматизировать рутинные проверки**.

**Регулярные ретроспективы по качеству кода**.

**24. Как учитывать пользовательский опыт (UX) в исследовании программного кода?**

**Анализ логов ошибок**.

**A/B-тестирование изменений**.

**Интеграция с аналитикой**.

**25. Какие навыки и знания необходимы для эффективного исследования программного кода?**

**Понимание алгоритмов и структур данных**.

**Опыт работы с инструментами анализа**.

**Знание best practices**.

**26. Какие примеры успешного анализа программного кода можно привести?**

**Linux Kernel** → постоянный аудит безопасности.

**Google Chromium** → автоматизированные тесты и fuzzing.

**27. Как анализ программного кода может повлиять на процесс разработки продукта?**

**Снижение количества багов в продакшене**.

**Ускорение разработки** за счёт лучшей поддерживаемости.

**28. В каких случаях стоит применять метод анализа кода к унаследованному ПО?**

**Перед рефакторингом**.

**При переходе на новые технологии**.

**29. Как применение современных технологий, таких как машинное обучение, может улучшить анализ кода?**

**Автоматическое исправление кода** (GitHub Copilot).

**Предсказание уязвимостей** на основе паттернов.

**30. Какие тенденции в исследовании программного кода наблюдаются на сегодняшний день?**

**Shift-left testing** (ранний анализ в CI/CD).

**AI-assisted code review**.

**Увеличение роли статического анализа**.