**1. Основные понятия**

1. Что такое отладчик и для чего он используется?

Отладчик — это инструмент для пошагового выполнения программы, анализа её состояния (значения переменных, регистров, стека) и поиска ошибок.

Используется для:

- Поиска багов (ошибок в коде).

- Анализа работы программы в реальном времени.

- Исследования чужого кода (в т. ч. без исходников).

2. В чем разница между отладчиком и дизассемблером?

- Отладчик позволяет запускать программу, ставить точки останова, менять значения переменных.

- Дизассемблер преобразует машинный код в ассемблер (без запуска программы).

3. Какие типы ошибок можно найти с помощью отладчика?

- Логические ошибки (неверные условия, циклы).

- Ошибки памяти (segmentation fault, утечки).

- Исключения (деление на ноль, доступ к nullptr).

- Непредвиденные состояния программы.

4. Что такое точка останова (breakpoint)?

Это метка, при достижении которой программа \*\*приостанавливает выполнение\*\*, позволяя изучить её состояние (переменные, стек вызовов).

**2. Работа с отладчиком**

5. Как запустить программу в режиме отладки в Visual Studio?

1. Откройте проект.

2. Нажмите F5 или выберите Debug → Start Debugging.

3. Для отладки без запуска: \*Debug → Attach to Process\*.

6. Основные команды отладчика

- Step Into (F11) — зайти внутрь функции.

- Step Over (F10) — выполнить текущую строку без входа в функции.

- Continue (F5) — продолжить выполнение до следующей точки останова.

- Step Out (Shift+F11) — выйти из текущей функции.

7. Как просмотреть значения переменных во время отладки?

- Локальные переменные: Окно Locals (автоматически отображает переменные текущей области видимости).

- Конкретные переменные: Навести курсор или добавить в Watch-окно.

8. Что такое Watch-окно? Как его использовать?

Watch — окно для отслеживания значений переменных/выражений в реальном времени.

- Добавьте переменную: Введите её имя в \*Watch\* (например, x или array[0]).

- Можно вычислять выражения: x + y \* 2.

9. Как отладить программу, если она падает без сообщения об ошибке?

1. Включите все исключения (Debug → Windows → Exception Settings).

2. Используйте отладку дампа памяти (если программа упала).

3. Проверьте логи и стек вызовов (Debug → Call Stack).

**3. Дизассемблирование и анализ кода**

10. Что такое дизассемблер и какие задачи он решает?

Дизассемблер преобразует машинный код в ассемблерный.

Задачи:

- Анализ работы программы без исходников.

- Поиск уязвимостей.

- Реверс-инжиниринг (например, взлом защиты).

11. Популярные дизассемблеры

- IDA Pro — профессиональный инструмент.

- Ghidra — бесплатный аналог от NSA.

- radare2 — для командной строки.

12. Как найти функцию main в дизассемблированной программе?

- В C/C++: Ищите вызов \_\_libc\_start\_main (аргумент — адрес main).

- В Windows PE: Проверьте точку входа (EntryPoint), затем ищите вызовы API.

13. Как дизассемблер помогает найти проверку пароля?

1. Ищите строки (например, "Password incorrect").

2. Найдите функцию, где сравниваются введённые данные с эталоном (инструкции cmp, je).

14. Как определить, какие библиотеки использует программа?

- Windows: ldd (Linux) или Dependency Walker.

- Linux: ldd <программа>.

- В дизассемблере: раздел .idata (импорты функций).

***4. Практическое применение***

15. Как найти ошибку "деление на ноль" с помощью отладчика?

1. Поставьте точку останова перед операцией деления.

2. Запустите программу (\*\*F5\*\*).

3. Проверьте значение делителя в \*Watch\*-окне.

4. Если делитель `0`, изучите, как он вычисляется.

5. Обратная разработка (Reverse Engineering)

16. Как восстановить алгоритм программы без исходного кода?

1. Дизассемблируйте программу (IDA/Ghidra).

2. Анализируйте ключевые функции (ввод/вывод, условия).

3. Восстановите логику по ассемблеру.

17. Как найти ключевые функции в дизассемблированной программе?

- Ищите вызовы API (например, printf, strcmp).

- Анализируйте \*\*строки\*\* (часто указывают на назначение функции).

18. Как определить, использует ли программа шифрование?

- Ищите функции с битовыми операциями (XOR, SHR, SHL).

- Проверьте вызовы криптобиблиотек (OpenSSL, CryptoAPI).

19. Как проанализировать вредоносное ПО?

1. Запустите в песочнице (VirtualBox, Cuckoo Sandbox).

2. Дизассемблируйте код.

3. Ищите подозрительные вызовы (например, CreateRemoteThread).

6. Дополнительные вопросы

20. Разница между статическим и динамическим анализом?

- Статический: Анализ кода без запуска (дизассемблирование).

- Динамический: Анализ во время выполнения (отладчик, трассировка).

21 Этические аспекты работы с дизассемблерами и отладчиками

- Можно: Анализ собственного ПО, исследование уязвимостей (с разрешения).

- Нельзя: Взлом лицензионного ПО, создание вредоносных программ.