**1. Назначение отладчика VC++**

Отладчик Visual C++ (VC++) предназначен для поиска и устранения ошибок в программном коде, анализа выполнения программы, контроля значения переменных и состояния программы в реальном времени.

**2. Основные возможности отладчика VC++**

* Установка и управление точками останова.
* Пошаговое выполнение программы (Step Into, Step Over, Step Out).
* Просмотр значений переменных, объектов и массивов.
* Исследование стека вызовов.
* Отображение ассемблерного кода программы.
* Работа с окнами памяти и регистров.
* Анализ многопоточных приложений.
* Поддержка профилирования и анализа производительности.

**3. Что такое точка останова?**

Точка останова (breakpoint) — это метка в коде, где выполнение программы останавливается во время отладки, позволяя разработчику анализировать текущее состояние программы, значения переменных и другие параметры.

**4. Способы установки точек останова**

1. **Клик в редакторе кода**: Щелчок на поле слева от строки кода.
2. **Контекстное меню**: ПКМ по строке кода → "Установить точку останова".
3. **Горячие клавиши**: Нажать F9 в строке кода.
4. **Меню Debug**: Выбрать Debug → Toggle Breakpoint.
5. **Условная точка останова**: Через Debug → New Breakpoint, с указанием условий остановки.

**5. Какие способы запуска отладчика вы знаете?**

1. **Запуск с отладкой**: F5 или Debug → Start Debugging.
2. **Запуск без отладки**: Ctrl+F5 или Debug → Start Without Debugging.
3. **Присоединение к процессу**: Debug → Attach to Process.

**6. Основные возможности управления режимами отладки**

* **Step Into (F11)**: Пошаговое выполнение программы с переходом внутрь вызовов функций.
* **Step Over (F10)**: Выполнение текущей строки без входа в вызовы функций.
* **Step Out (Shift+F11)**: Завершение выполнения текущей функции и возврат к вызову.
* **Continue (F5)**: Продолжение выполнения программы до следующей точки останова.
* **Pause (Ctrl+Alt+Break)**: Приостановка выполнения программы.
* **Restart (Ctrl+Shift+F5)**: Перезапуск отладки.

**7. Способы отображения значений переменных**

1. **Окно Watch**: Ручное добавление переменных для мониторинга.
2. **Окно Locals**: Автоматический показ локальных переменных.
3. **Окно Autos**: Автоматический показ используемых переменных.
4. **Инструментальные подсказки**: Наведение мыши на переменную в коде.
5. **Окно Immediate**: Запрос значений и выполнение выражений вручную.
6. **Окно Call Stack**: Просмотр переменных в рамках стека вызовов.

**8. Назначение отладочных окон Debuggerа**

Отладочные окна служат для анализа состояния программы:

* **Call Stack**: Показывает стек вызовов.
* **Locals**: Отображает локальные переменные текущей функции.
* **Autos**: Показывает автоматически определенные переменные.
* **Watch**: Позволяет вручную отслеживать переменные.
* **Memory**: Исследует содержимое памяти.
* **Registers**: Отображает значения регистров процессора.
* **Disassembly**: Показывает ассемблерный код программы.

**9. Для каких целей используется окно памяти?**

Окно памяти позволяет:

* Просматривать и изменять содержимое памяти по заданному адресу.
* Анализировать структуру данных, таких как массивы или указатели.
* Диагностировать ошибки доступа к памяти (например, при работе с указателями).

**10. Для каких целей используется окно регистров?**

Окно регистров отображает значения регистров процессора.  
Используется для:

* Анализа работы программы на уровне машинных инструкций.
* Отладки низкоуровневого кода (например, системного или встроенного ПО).
* Отслеживания изменений регистров в ходе выполнения инструкций.

**11. Для каких целей используется окно дизассемблированного кода?**

Оно используется для:

* Просмотра ассемблерного кода, соответствующего выполняемому C++ коду.
* Отладки участков программы, где нет исходного кода.
* Анализа работы оптимизирующего компилятора.

**12. Что отражается в окне дизассемблированного кода?**

* Ассемблерные инструкции, выполняемые процессором.
* Адреса памяти, соответствующие инструкциям.
* Значения переменных и регистров, связанные с текущими командами.
* Интеграция с исходным кодом (если доступен), показывая соответствующие строки кода.