1. Поясните различие между статической библиотекой и DLL.

- Статическая библиотека: включается в исполняемый файл во время компиляции. Функции из такой библиотеки становятся частью исполняемого файла, и библиотека больше не требуется после компиляции.

- DLL (динамически подключаемая библиотека): подключается к программе во время выполнения. Библиотека существует отдельно от исполняемого файла и может быть обновлена или заменена без необходимости перекомпиляции программы.

2. Назовите преимущества и недостатки тех и других библиотек.

- Статические библиотеки:

- Преимущества: Простота использования, нет необходимости в дополнительных файлах при распространении программы.

- Недостатки: Увеличение размера исполняемого файла, невозможность обновления библиотеки без перекомпиляции программы.

- DLL:

- Преимущества: Возможность обновления и замены без перекомпиляции программы, уменьшение размера исполняемого файла, общие библиотеки могут использоваться несколькими программами одновременно.

- Недостатки: Необходимость наличия библиотеки на устройстве пользователя, сложности с управлением зависимостями.

3. Как к проекту подключается статическая библиотека?

- Статическая библиотека подключается к проекту на этапе компиляции. Необходимо добавить соответствующий файл `.lib` в проект и указать компилятору использовать его. В коде проекта нужно включить заголовочный файл библиотеки.

4. Как к проекту подключается DLL?

- DLL подключается к проекту на этапе выполнения. Необходимо использовать функции `LoadLibrary` для загрузки DLL и `GetProcAddress` для получения адресов функций. В коде проекта нужно включить заголовочный файл библиотеки, если функции DLL должны быть вызваны.

5. Назовите 4 вида соглашений о вызове функций (методов). Какими модификаторами они обозначаются?

- \*\*cdecl\*\*: `\_\_cdecl`

- \*\*stdcall\*\*: `\_\_stdcall`

- \*\*fastcall\*\*: `\_\_fastcall`

- \*\*thiscall\*\*: используется по умолчанию для методов класса в C++.

6. Что такое декорация имен? Как ее избежать для экспорта?

- Декорация имен (name mangling)\*\*: процесс преобразования имен функций и переменных для их уникальности в языке программирования (обычно в C++). Это может включать добавление информации о типах аргументов и классов.

- Избежать декорации: Для экспорта функций в C++ можно использовать конструкцию `extern "C"`, что отключает декорацию имен.

7. Как указывается экспортируемая (импортируемая) функция класс?

- Используются макросы `\_\_declspec(dllexport)` для экспорта и `\_\_declspec(dllimport)` для импорта функций или классов. Пример:

```cpp

#ifdef BUILDING\_DLL

#define DLL\_EXPORT \_\_declspec(dllexport)

#else

#define DLL\_EXPORT \_\_declspec(dllimport)

#endif

class DLL\_EXPORT MyClass {

public:

void MyMethod();

};

```

8. В чем различие между явным и неявным связыванием?

- \*\*Неявное связывание\*\*: библиотека подключается автоматически при запуске программы. Необходимые зависимости указываются на этапе компиляции.

- \*\*Явное связывание\*\*: библиотека загружается и функции из нее вызываются в явном виде во время выполнения программы с помощью функций `LoadLibrary` и `GetProcAddress`.

9. Назовите порядок поиска DLL.

- Порядок поиска DLL обычно следующий:

1. Каталог, из которого загружено приложение.

2. Текущий рабочий каталог.

3. Системный каталог Windows (`C:\Windows\System32`).

4. Каталог Windows (`C:\Windows`).

5. Каталоги, указанные в переменной окружения `PATH`.

10. Для чего нужна функция DllMain?

- Функция `DllMain` выполняет инициализацию и очистку DLL. Она вызывается при различных событиях, таких как загрузка DLL в память, выгрузка DLL из памяти, создание или завершение потока.

11. Что называется переадресацией вызовов?

- \*\*Переадресация вызовов (import redirection)\*\*: механизм, позволяющий перенаправить вызовы функций из одной DLL в другую. Это используется для обновления библиотек или исправления ошибок.

12. Какие DLL называются известными?

- Известные DLL (well-known DLLs) - это библиотеки, которые предустановлены в операционной системе и доступны для всех приложений, например, `kernel32.dll`, `user32.dll`.

13. Для чего нужна команда dumpbin?

- Команда `dumpbin` используется для просмотра содержимого различных файлов, таких как исполняемые файлы и библиотеки. Она может показывать таблицы экспорта и импорта, заголовки файлов и другую информацию.

14. Для чего нужна команда rebase?

- Команда `rebase` используется для изменения базового адреса DLL, чтобы избежать конфликтов с другими загружаемыми DLL. Это улучшает производительность за счет уменьшения необходимости перезагрузки DLL в разные адресные пространства.

15. Создайте статическую библиотеку regina.lib и скомпонуйте ее с законченным приложением regina.

- Статическая библиотека должна содержать функцию `GetReginaRecordLib`, которая возвращает новую запись `Regina Record`. Приложение `regina` должно вызывать эту функцию через пункт меню.

16. Создайте DLL regina.dll с одной функцией GetReginaRecordDll и подключите ее к приложению regina.

- Динамическая библиотека должна содержать функцию `GetReginaRecordDll`, которая возвращает новую запись `Regina Record`. Приложение `regina` должно вызывать эту функцию через пункт меню, используя неявное связывание.