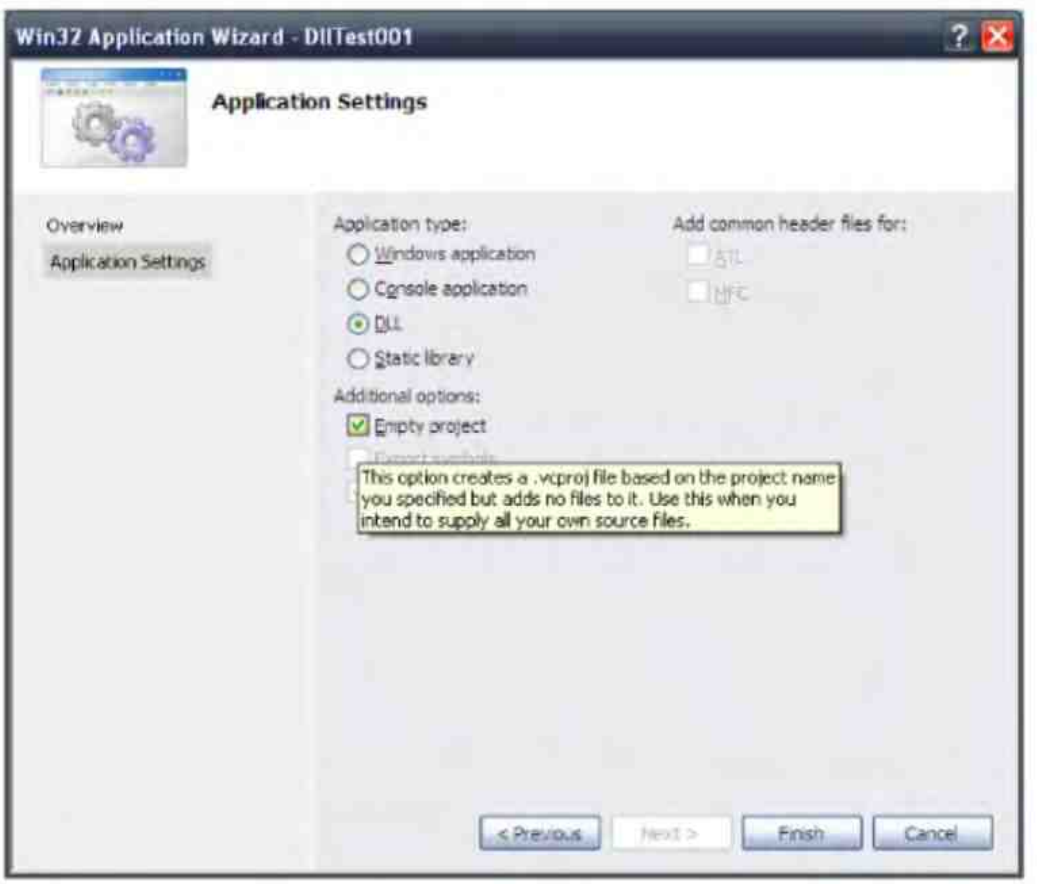
Лабораторная работа № 10
Динамически подключаемые библиотеки
(2 часа)

Содержание: Концепция DLL. Структура и разработка DLL. Неявная и явная загрузка DLL. Отложенная загрузка DLL.

Цель: изучить основы создания динамически линкуемых библиотек, варианты их загрузки и использования в разработке программного обеспечения; получить навыки создания, статической и динамической загрузки и выгрузки DLL, описания файла определения модуля.

Одним из применений концепции отображения файлов в память являются DLL – динамически линкуемые библиотеки. Основные идеи, положенные в основу создания DLL – это возможность совместного использования кода различными приложениями, а также уменьшение объема используемой физической памяти при одновременной работе нескольких приложений, использующих одни и те же библиотеки. В этом случае один экземпляр DLL (раздел кода) проецируется в адресное пространство всех использующих его процессов. При этом каждый из процессов имеет собственную копию раздела данных динамически линкуемой библиотеки.

Создание динамической библиотеки в Microsoft Visual Studio незначительно отличается от создания проекта консольного приложения:



Точкой входа в DLL является функция DllMain:

```
BOOL WINAPI DllMain(
    HINSTANCE hinstDll,
    DWORD fdwReason,
    LPVOID lpvReserved
    );
```

Обратите внимание, функция возвращает логическое значение: TRUE в случае успешной загрузки и FALSE в случае ошибки.

В теле функции DllMain как правило выполняется выделение и освобождение ресурсов при загрузке и выгрузке DLL.

Кроме функции DllMain в динамически линкуемой библиотеке обычно могут быть объявлены как используемые только внутри библиотеки, так и **экспортируемые** функции и данные. Для объявления экспорта используется квалификатор:

```
__declspec (dllexport)
```

Для отключения декорирования имен описание экспортируемых символов обычно сопровождается указанием на генерацию имен в стиле C:

```
extern "C"
```

Объявлять и описывать экспортируемые символы можно в основном файле динамической библиотеки (содержащем функцию DllMain), но чаще всего объявления и определения символов выносятся в отдельный модуль с собственным заголовочным файлом.

Рассмотрим простейший пример:

```
extern "C" __declspec(dllexport) char Message[]="Hello,
World!";
extern "C" __declspec(dllexport) void SayMessage(char*
message)
{
        cout << "DLL say> " << message << endl;
        return;
}
```

В этом примере определяется функция void SayMessage(char*) и объявляется текстовая строка Message, которые экспортируются из динамической библиотеки.

После сборки проекта динамически линкуемой библиотеки Вы получите файл с расширением *.dll, содержащий саму библиотеку и файл с расширением *.lib, содержащий информацию, необходимую для импорта из динамической библиотеки.

Задание 1 (Разминка). Создайте в Microsoft Visual C++ проект динамически линкуемой библиотеки и соберите на основании приведенных выше фрагментов файл динамической библиотеки.

В случае, когда программа использует код или данные из динамической библиотеки, говорят, что она **импортирует** эти функции или данные.

Для загрузки динамически линкуемой библиотеки существует несколько основных способов: динамическая (явная) загрузка, статическая (неявная) загрузка и отложенная загрузка.

Для динамической загрузки используются функции `LoadLibrary` или `LoadLibraryEx`. Прототип функции `LoadLibrary` имеет следующий вид:

```
HMODULE LoadLibrary(
    LPCTSTR lpFileName
);
```

Здесь параметр `lpFileName` задает имя загружаемой динамической библиотеки. В случае отсутствия расширения подразумевается расширение DLL. Если указан неполный путь, то будет произведен поиск в следующем порядке:

1) каталог, из которого запущена использующая DLL программа;
2) системный каталог;
3) системный каталог 16-битных приложений;
4) каталог Windows;
5) текущий для запущенной программы каталог;
6) каталоги, перечисленные в переменной окружения `PATH`.

Если обнаружено, что библиотека с таким именем уже загружена, то будет использована именно она, без загрузки дополнительной копии, иначе библиотека будет загружена в оперативную память. В обоих случаях будет выполнена `DllMain` с параметром `fdwReason` равным `DLL_PROCESS_ATTACH`.

После того, как библиотека станет не нужна, ее можно отключить от вызвавшего процесса вызовом `FreeLibrary`:

```
BOOL FreeLibrary(
    HMODULE  hModule
);
```

В случае неудачи функция возвращает `FALSE`, успеха – `TRUE`. При этом ОС вызывает `DllMain` библиотеки с параметром `fdwReason` равным `DLL_PROCESS_DETACH`.

После загрузки библиотеки можно получить адрес экспортируемой из библиотеки функции с помощью функции `GetProcAddress`:

```
FARPROC GetProcAddress(
    HMODULE hModule,
    LPCSTR   lpProcName
);
```

В случае успешного завершения функция возвращает адрес экспортируемой из библиотеки функции, в случае неудачи – `NULL`.

Параметр `hModule` при вызове должен содержать дескриптор загруженной DLL; параметр `lpProcName` – имя импортируемой в программу

функции или ее номер. Для задания номера функции легче всего использовать макрос MAKEINTRESOURCE(N), где N должно задавать индекс функции.

Рассмотрим пример динамической загрузки DLL:

```
HMODULE hMyDll;
    void (*Say)(char*);
    char* word;

    hMyDll = LoadLibrary(L"SimpleDLL");
    if(hMyDll == NULL) { wcerr << L"error loading library"
<< endl; system("pause"); return 3;}

    word = (char*)GetProcAddress(hMyDll, "Message");
    if(!word){ wcerr << L"error loading message form
library" << endl; system("pause"); return 4;}

    Say = (void(*)(char*))GetProcAddress(hMyDll,
"SayMessage");
    if(!Say){ wcerr << L"error loading function from
library" << endl; system("pause"); return 5;}

    Say(word);
    system("pause");
   FreeLibrary(hMyDll);
```

Задание 2 (Разминка). Создайте консольное приложение, использующее созданную в задании 1 библиотеку. Проверьте его работоспособность.

Задание 3. На основе 2 задания лабораторной работы №5 «Управление файлами и каталогами» разработайте динамическую библиотеку для извлечения по указанной позиции текстовой строки указанной длины из файла. Продемонстрируйте использование этой функции:

a) при помощи неявной загрузки;
b) при помощи явной загрузки.