Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Чепоков Елизар Сергеевич

# БИБЛИОТЕКИ

*Реферат*

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Доцент кафедры
информационных
технологий в бизнесе
_____
Л. Н. Лядова

Пермь, 2020 год

**DLL** (Dynamic Link Library) – «библиотека динамической компоновки» или «динамически подключаемая библиотека» динамическая библиотека, позволяющая многократное использование различными программными приложениями. Каждая программа может использовать возможности, содержащиеся в библиотеке для реализации определенных задач. Это помогает стимулировать повторное использование кода и эффективное использование памяти.
С помощью DLL программу можно модулировать на отдельные компоненты. Каждый модуль может быть загружен в основную программу во время выполнения, если этот модуль установлен. Поскольку модули разделены, время загрузки программы быстрее, а модуль загружается только при запросе этой функциональности.

Существуют два основных способа подключить DLL к программе - **явный и неявный**.

При **неявном** подключении (implicit linking) линкеру передается *библиотека импорта* (обычно имеет расширение lib), содержащая список переменных и функций DLL, которые могут использовать приложения. Обнаружив, что программа обращается хотя бы к одной из них, линкер добавляет в целевой exe-файл *таблицу импорта*. Таблица импорта содержит список всех DLL, которые использует программа, с указанием конкретных переменных и функций, к которым она обращается. Позже, когда exe-файл будет запущен, загрузчик проецирует все DLL, перечисленные в таблице импорта, на адресное пространство процесса; в случае неудачи весь процесс немедленно завершается.

При **явном** подключении (explicit linking) приложение вызывает функцию LoadLibrary, чтобы загрузить DLL, затем использует функцию GetProcAddress, чтобы получить указатели на требуемые функции (или переменные), а по окончании работы с ними вызывает FreeLibrary, чтобы выгрузить библиотеку и освободить занимаемые ею ресурсы.

Для того чтобы привести примеры явного и неявного подключения библиотеки необходимо создать саму библиотеку.

## Создание библиотеки

Необходимо создать проект DLL в Visual Studio.

Код библиотеки MyLibrary приведен ниже.

```
#include "stdafx.h"
#include <iostream>

extern "C" __declspec(dllexport) double Plus(double a, double b) {
	return a + b;
}
extern "C" __declspec(dllexport) double Minus(double a, double b) {
	return a - b;
}
extern "C" __declspec(dllexport) const char* HelloWorld(void) {
	return "hello DLL!";
}
```

__declspec(dllexport) – ключевое слово для объявления функции.

Файл заголовка:

```
#pragma once

#ifdef MATHLIBRARY_EXPORTS
#define MATHLIBRARY_API __declspec(dllexport)
#else
#define MATHLIBRARY_API __declspec(dllimport)
#endif

extern "C" MATHLIBRARY_API void library();
```

Для завершения работы необходимо **собрать** данную библиотеку, выбрав одноименный пункт.

## Явное подключение

Динамическое (явное) подключение предполагает, что библиотека подключается в приложение в момент его исполнения. Для этого необходимо использовать функции библиотеки windows.h

GetProcAddress – извлекает адрес экспортируемой функции или переменной из указанной библиотеки.

LoadLibrary – загружает указанный модуль (дескриптор) в адресное пространство вызывающего процесса.

Дескриптор необходимо сохранить в переменной, так как он будет использоваться всеми остальными функциями, предназначенными для работы с DLL. После загрузки бибилиотеки, адрес любой из содержащихся в ней функций можно получить с помощью GetProcAddress, которой необходимо передать дескриптор библиотеки и имя функции. Для возвращаемого значения функции GetProcAddress нужно определить новые имена используемым в библиотеке типам

при помощи ключевого слова typedef, с помощью них будет удобнее взаимодействовать с функциями библиотеки. Выглядит это следующим образом:

```
typedef double(*PlusFunc) (double, double);
typedef double(*MinusFunc) (double, double);
typedef char*(*HelloWorldFunc)(void);
```

Далее необходимо загрузить библиотеку, получить из нее функции Plus, Minus и HelloWorld и вызвать их, передав в функцию Mathematics параметра типа double. Стоит также выводить сообщения об успешной или неуспешной загрузке библиотеки и функции. Код явного подключения представлен ниже.

```
setlocale(LC_ALL, "Russian");

HMODULE hLib = LoadLibrary(L"RefDll.dll");
if (hLib) {
	cout << "Библиотека загружена." << endl;
	PlusFunc plus = (PlusFunc)GetProcAddress(hLib, "Plus");
	if (plus) {
		double result = plus(100.5, 50.5);
		cout << result << endl;
	}
	else if (minus) {
		double result = minus(100.5, 50.5);
		cout << result << endl;
	}
	else {
	HelloWorldFunc foo = (HelloWorldFunc)GetProcAddress(hLib, "HelloWorld");
	if (foo) cout << foo() << endl;
	FreeLibrary(hLib);
	}
else
	cout << "Библиотека не найдена." << endl;
```

## Неявное подключение

Статическое (неявное) подключение DLL к приложению – более простой метод подключения. Для этого необходимо добавить в проект в папку «Файлы ресурсов» скомпилированный lib-файл библиотеки и в файле исходного кода, где будут вызываться функции библиотеки, необходимо поместить прототип импортируемых функций. Результат добавления lib-файла представлен на рисунке

Код прототипов импортируемых функций и вызов этих функции в методе main представлен ниже.

```
extern "C" __declspec(dllimport) const char* HelloWorld(void);
extern "C" __declspec(dllimport) double Plus(double, double);
extern "C" __declspec(dllimport) double Minus(double, double);

int main()
{
	cout << HelloWorld() << endl;
	cout << Plus(100.5, 50.5) << endl;
	cout << Minus(100.5, 50.5) << endl;
	return 0;
}
```