Лабораторная работа "Разработка и применение динамически загружаемых библиотек в Win32 приложения"

Задание

1. Изучить технологию создания и использования динамически загружаемых библиотек в WIN32 API.

Источники информации:

1	Джеффри Рихтер.Windows для профессионалов./ Пер. с англ. -М.:Издательский отдел "Русская редакция" ТОО "Channel Trading Ltd.", 1995. и более поздние издания
2	Саймон Р. Windows 2000 API. Энциклопедия программиста: Пер. с англ./ Ричард Саймон — К.: Издательство "ДиаСофт", 2001. -1088 966-7393-74-7
3	Круглински Д., Уингоу С., Шеферд Дж.Программирование на Microsoft Visual C++ для профессионалов/ Пер с англ.- СПб: Питер; М.: Издательско-торговый дом "Русская редакция", 2001, — 864 с.: ил. ISBN 5-272-00385-3, УДК 004.43, ББК 32.973-26 018
4	Румянцев И.В. Азбука программирования в Win32 API. 2-е изд. стереотип.- М.: Радио и связь, Горячая линия-Телеком, 1999ю-272с. : ил. ISBN 5-256-01491-9.
5	MSDN \ Platform SDK \ Base Services \ Dynamic-Link Library
6	MSDN \ Visual Studio 6.0 Documentation \ Visual C++ Documentation \ Using Visual C++ \ Visual C++ Programmer Guide \ Adding Programm Functionality \ Details \ DLL Topic

2. Создать динамически загружаемую библиотеку, экспортирующую функции и глобальные переменные в соответствии с вариантом задания.

Проект и библиотеку назвать **< FAMILIA>_DLL** (в названии проекта присутствует ваша фамилия в латинской транскрипци).

В функции входа библиотеки DllMain предусмотреть для каждой из возможных причин вызова вывод соответствующих сообщений. Например, для причины DLL_PROCESS_ATTACH вывести сообщение вида "Загружается библиотека ИМЯ_МОДУЛЯ. Проект ФАМИЛИЯ_АВТОРА" .

В таблице вариантов заданы прототипы экспортируемых функций. Содержимое каждой из функций предлагается определить самостоятельно.

Глобальная переменная **g_nDllCallsCount** должна использоваться в качестве счетчика загрузок библиотеки. Ее значение наращивается на единицу всякий раз, когда какой-либо процесс загружает библиотеку (переменная, разделяемая между процессами).

Глобальная переменная **g_nFnCallsCount** должна использоваться в качестве счетчика вызова функций библиотеки в отдельном процессе. Ее значение увеличивается клиентской программой - пользователем библиотеки.

3. Создать клиентское приложение < **FAMILIA>_DLL_I** с неявным подключением библиотеки.

Применить технологию неявного подключения библиотеки (Implicit Linking). В приложении выполнить обращение к функциям и переменным библиотеки. Обращение к функциям демонстривать выводом значений аргументов вызова функций, полученного результата и значений глобальных переменных библиотеки (счетчика вызова функций и счетчика загрузок библиотеки). Просмотреть список экспорта библиотеки с помощью DUMPBIN.EXE.

4. Создать клиентское приложение <**FAMILIA>_DLL_E** с явным подключением библиотеки.

Просмотреть список экспорта библиотеки с помощью DUMPBIN.EXE. Добавить в проект библиотеки файл .DEF для запрета декорирования имен и перекомпилировать проект. Просмотреть список экспорта библиотеки с помощью DUMPBIN.EXE.

Создать клиентское приложение с использованием явного подключения библиотеки (Explicit Linking) . В приложении выполнить обращение к функциям и переменным библиотеки. Вывод должен содержать значения аргументов вызова функций, полученный результат и значения глобальных переменных библиотеки (счетчик вызова функций и счетчик загрузок библиотеки в разных процессах). Попробовать выполнить вызов функций и переменых библиотеки с использованием ординала (hint).

5. Оформить отчет о работе.

6. Выполнить задание для самостоятельной работы и оформить отчет.

(задания для самостоятельной работы в отдельном файле)

Варианты заданий.

Вариант	**Функции**	**Переменные**
1	int WINAPI **Fun11** (int, int) float **Fun12** (int, int, int) void **Fun13** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
2	float **Fun21**(float, int) int WINAPI **Fun22** (int, int) void **Fun23** (double in, double *out)	g_nDllCallsCount(разделяемая) g_nFnCallsCount

3	int WINAPI **Fun31** (double, double) float **Fun32** (int, int, int) void **Fun33** (int in, int *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
4	float **Fun41**(int, int) double WINAPI **Fun42** (int) void **Fun43** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
5	double **Fun51** (int, int) int WINAPI **Fun52** (int, int, int) void **Fun53** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
6	float **Fun61**(int, int) float WINAPI **Fun62** (int, int, int) void **Fun63** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
7	int WINAPI **Fun71**(int, int) double **Fun72** (int, int) void **Fun73** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
8	float **Fun81**(int, int) int WINAPI **Fun82** (int, int, int) void **Fun83** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
9	int **Fun91** (int, int) float WINAPI **Fun92** (float, float) void **Fun93** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
10	int WINAPI **Fun101** (int, int) float **Fun102** (int, int, int) void **Fun103** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
11	float **Fun111**(float, int) int WINAPI **Fun112** (int, int) void **Fun113** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount

12	int WINAPI **Fun121** (double, double) float **Fun122** (int, int, int) void **Fun123** (int in, int *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
13	float WINAPI **Fun131**(int, int) double **Fun132** (int) void **Fun133** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
14	double **Fun141** (int, int) int WINAPI **Fun142** (int, int, int) void **Fun143** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount
15	int WINAPI **Fun151**(int, int) float **Fun152** (int, int, int) void **Fun153** (double in, double *out)	g_nDllCallsCount (разделяемая) g_nFnCallsCount