

# WinAPI + DLL

Адамчук Денис Викторович

20.11.2020

CQG

# Структура лекции

- WinAPI**
  - Platform SDK
  - Объекты ядра
  - Описатели объектов ядра
  - «Жизненный цикл» объекта
- Пользовательский интерфейс**
  - Окна
  - Взаимодействие с пользователем
  - Оконные сообщения
  - Архитектура обработки сообщений
    - Очередь
    - Отправка/обработка
- DLL**
  - Предыстория
  - Жизненный цикл разработки
  - Виды использования
  - Цели использования
  - Адресное пространство
  - Экспортируемые функции
  - Функция входа/выхода
- Использование DLL в коде**
  - Алгоритм загрузки библиотек
  - Неявное связывание
  - Явное связывание
  - Искажение имен
- Недостатки DLL**

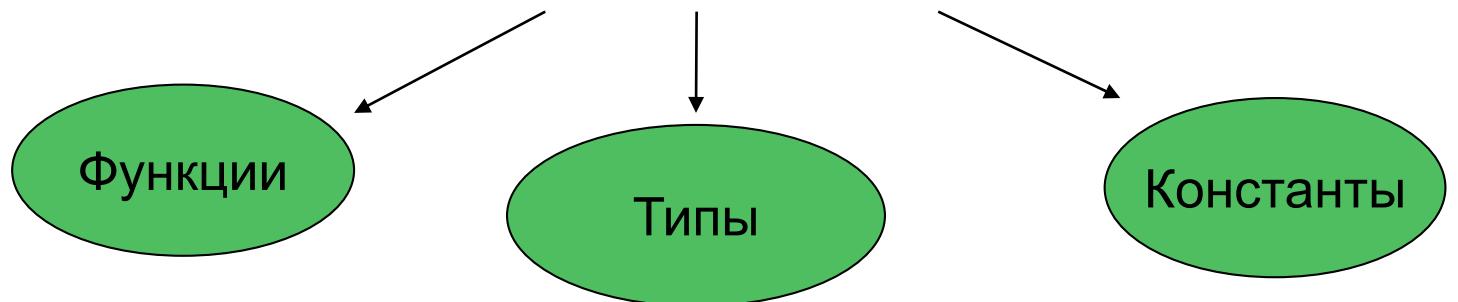
# WinAPI

## □ WinAPI

- Platform SDK
- Объекты ядра
- Описатели объектов ядра
- «Жизненный цикл» объекта

# WinAPI: Состав

## Win32 Application Programming Interface



C++ Standard Library  
.NET Framework

MFC (Microsoft Foundation Classes)

# WinAPI: Platform SDK (Software Development Kit)

Platform SDK - описание WinAPI

Состав:

- Библиотеки импорта
- Заголовочные файлы
- Примеры
- Документация

Установка:

- Входит в поставку Visual Studio
- Доступно отдельно (бесплатно)

Подобласти WinAPI:

- **Kernel objects**
- Memory
- UI
- Network
- Security
- I/O
- **DLL**
- ...

# WinAPI: Объекты ядра



Process

Thread

Event

Mutex

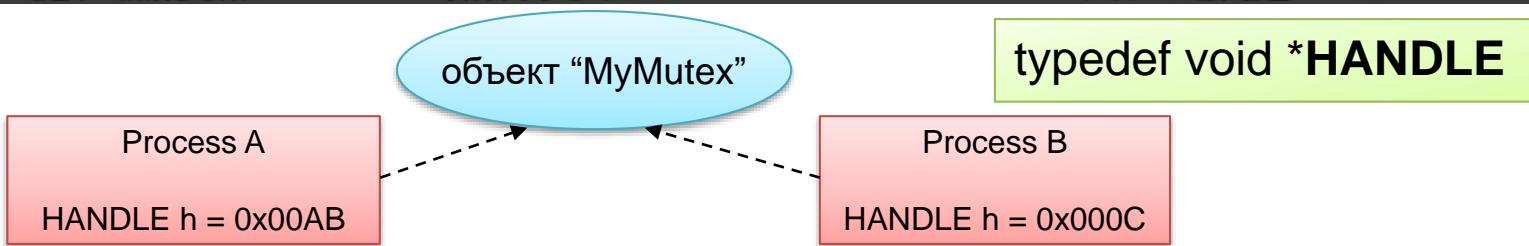
Semaphore

File

CreateEvent(...)  
CreateMutex(...)  
CreateProcess(...)  
CreateThread(...)

Для именованных объектов:  
OpenEvent(...)  
OpenMutex(...)  
OpenProcess(...)  
OpenThread(...)

# WinAPI: Описатели объектов ядра



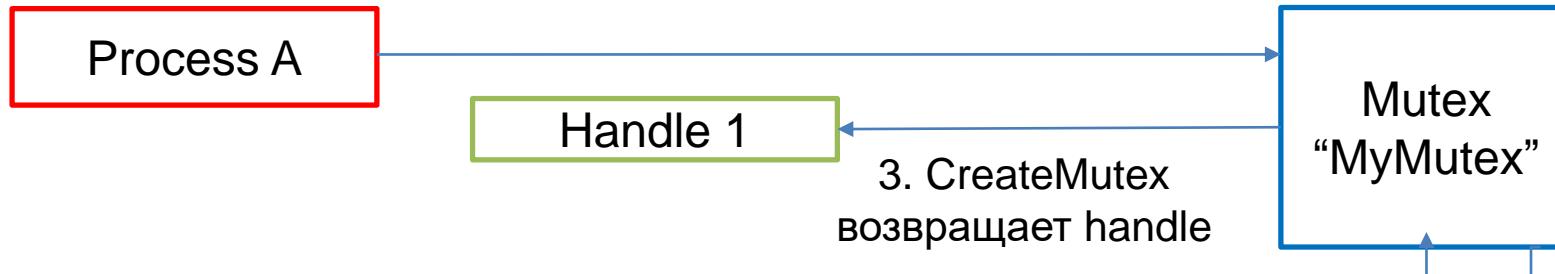
```
void MyClassA::createMutex()
{
    HANDLE h =
        CreateMutex(0, FALSE, "MyMutex");
    assert( 0 != h );
    mutex_ = h;
}

void MyClassA::releaseMutex()
{
    CloseHandle(mutex_);
}
```

```
void MyClassB::getAndUseMutex()
{
    HANDLE h =
        OpenMutex(0, FALSE, "MyMutex");
    if ( 0 == h )
    {
        // ... some handling
    }
    // use mutex here
    CloseHandle(h);
}
```

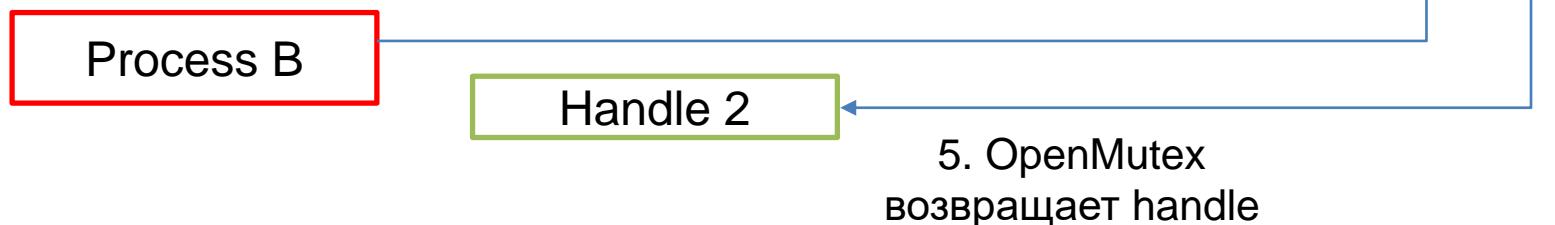
# WinAPI: «Жизненный цикл» объекта: Создание

1. Вызов CreateMutex



2. CreateMutex создаёт mutex в памяти

4. Вызов OpenMutex



<https://docs.microsoft.com/en-us/windows/desktop/sysinfo/kernel-objects>

# WinAPI: «Жизненный цикл» объекта: Удаление

1. Вызов CloseHandle



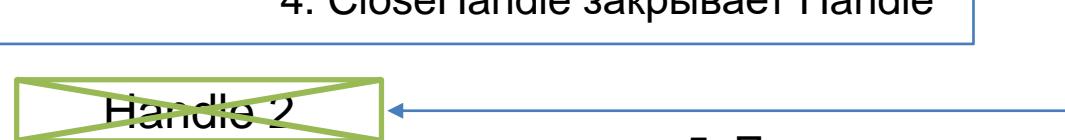
2. CloseHandle закрывает Handle



3. Вызов OpenMutex



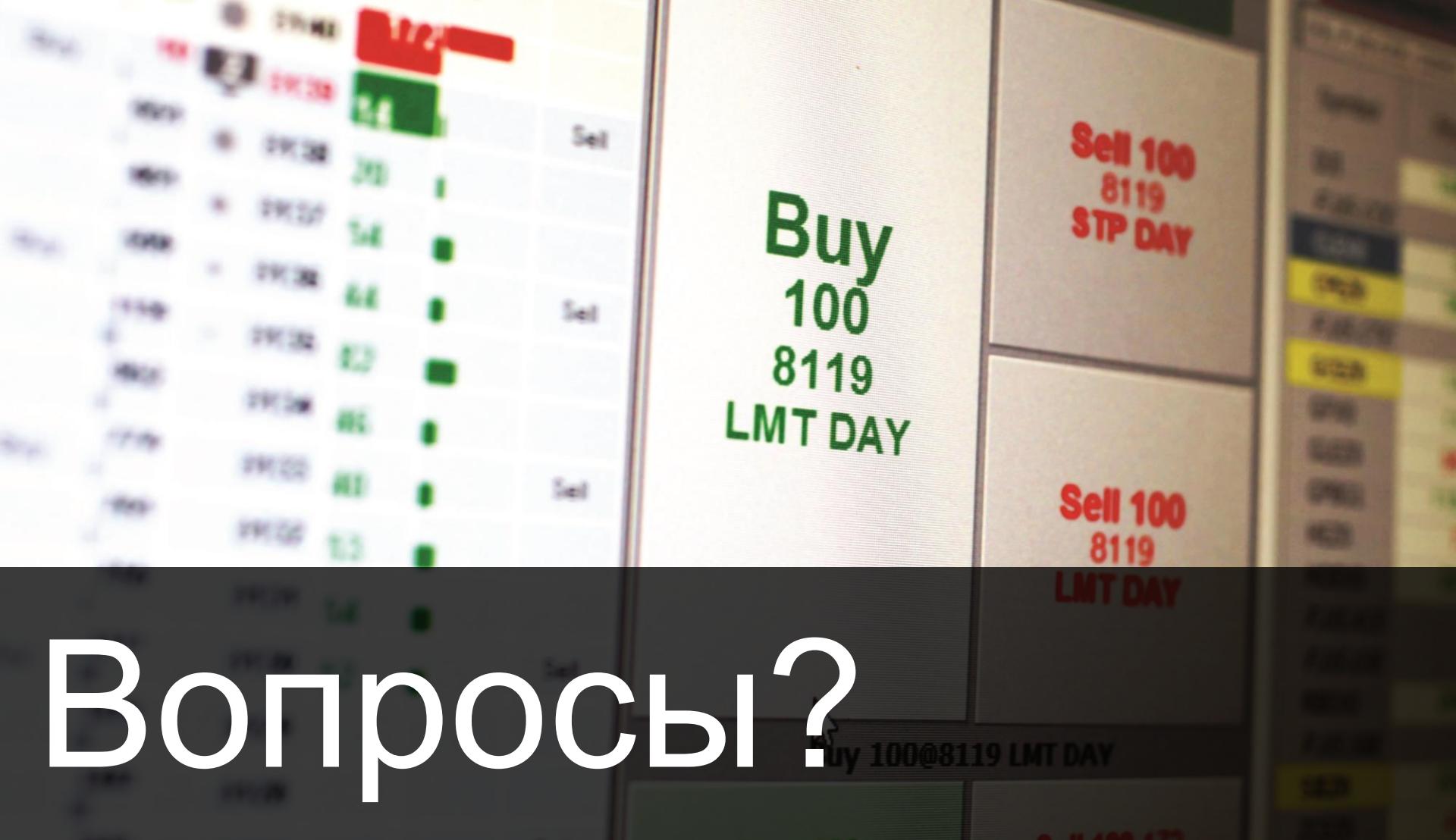
4. CloseHandle закрывает Handle



5. После закрытия последнего handle на объект, ОС удаляет его из памяти

<https://docs.microsoft.com/en-us/windows/desktop/sysinfo/kernel-objects>

# Вопросы?

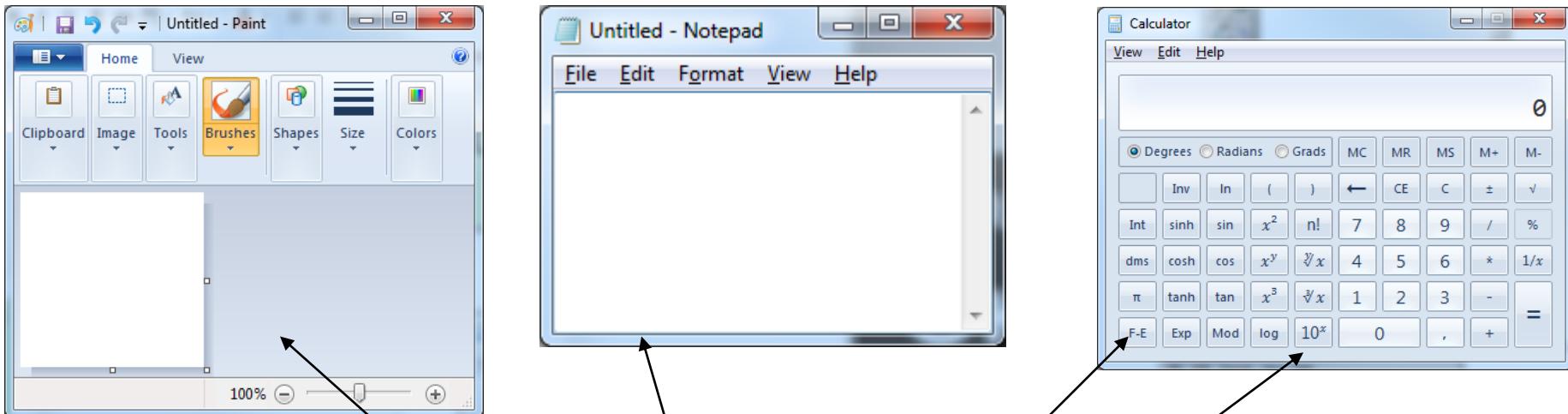


# Пользовательский интерфейс

## □ Пользовательский интерфейс

- Окна
- Взаимодействие с пользователем
- Оконные сообщения
- Архитектура обработки сообщений
  - Очередь
  - Отправка/обработка

# Пользовательский интерфейс: Окна



- USER objects
- Properties
- Styles
- Parent/Child/Sibling

- Уникальный идентификатор  
окна (дескриптор)

# Пользовательский интерфейс: Взаимодействие с пользователем

## Пакетный подход (Batch programming)

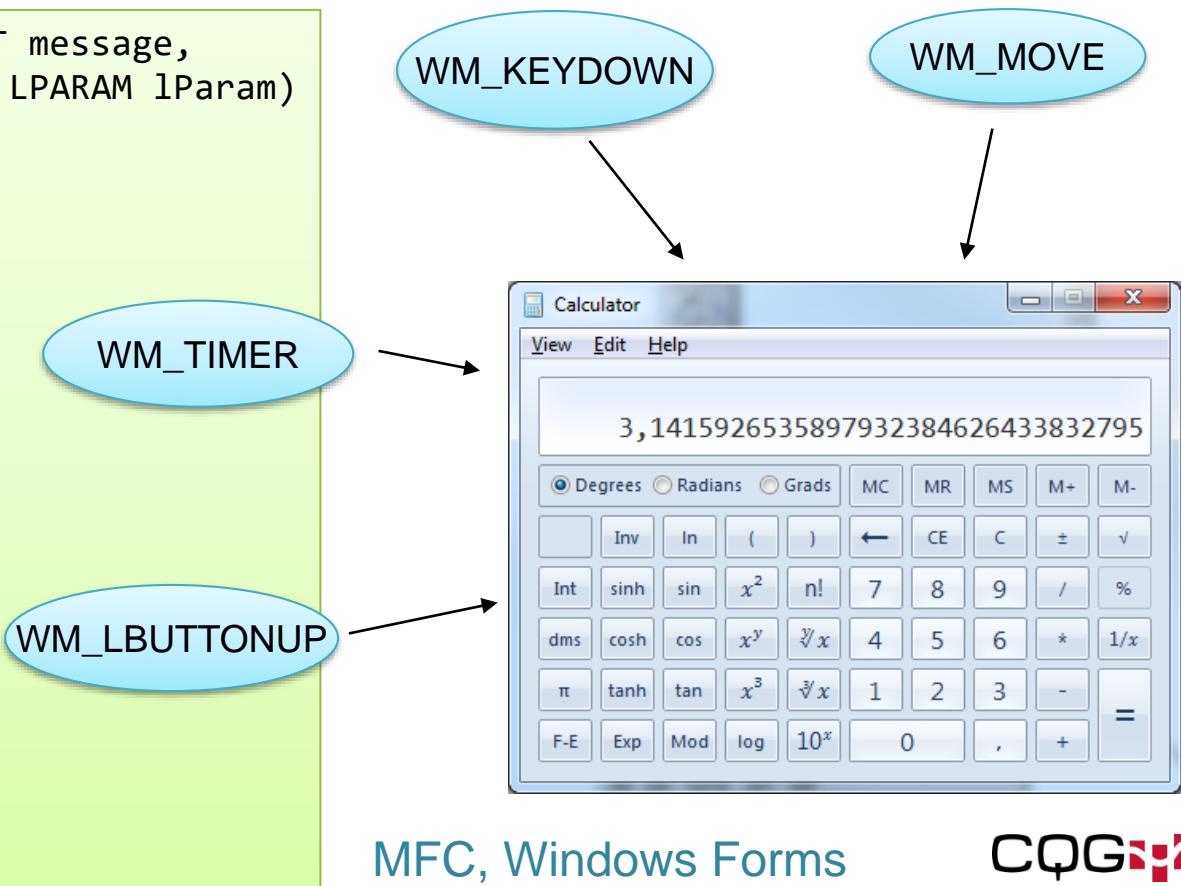
```
int a;  
int b;  
  
printf("Type a\n");  
scanf("%d", &a);  
printf("Type b\n");  
scanf("%d", &b);  
  
printf("a + b = %d\n", a + b);
```

## Событийно-ориентированный подход (Event-driven programming)

```
void processEvent(Event e);  
  
while (true) {  
    const Event e = get_event();  
    processEvent(e);  
}  
  
void processEvent(Event e) {  
    switch (e.type) {  
        case keyPressed: /* process keyboard */ break;  
        case mouseClick: /* process mouse */ break;  
        case timer: /* process timer */ break;  
    }  
}
```

# Пользовательский интерфейс: Оконные сообщения

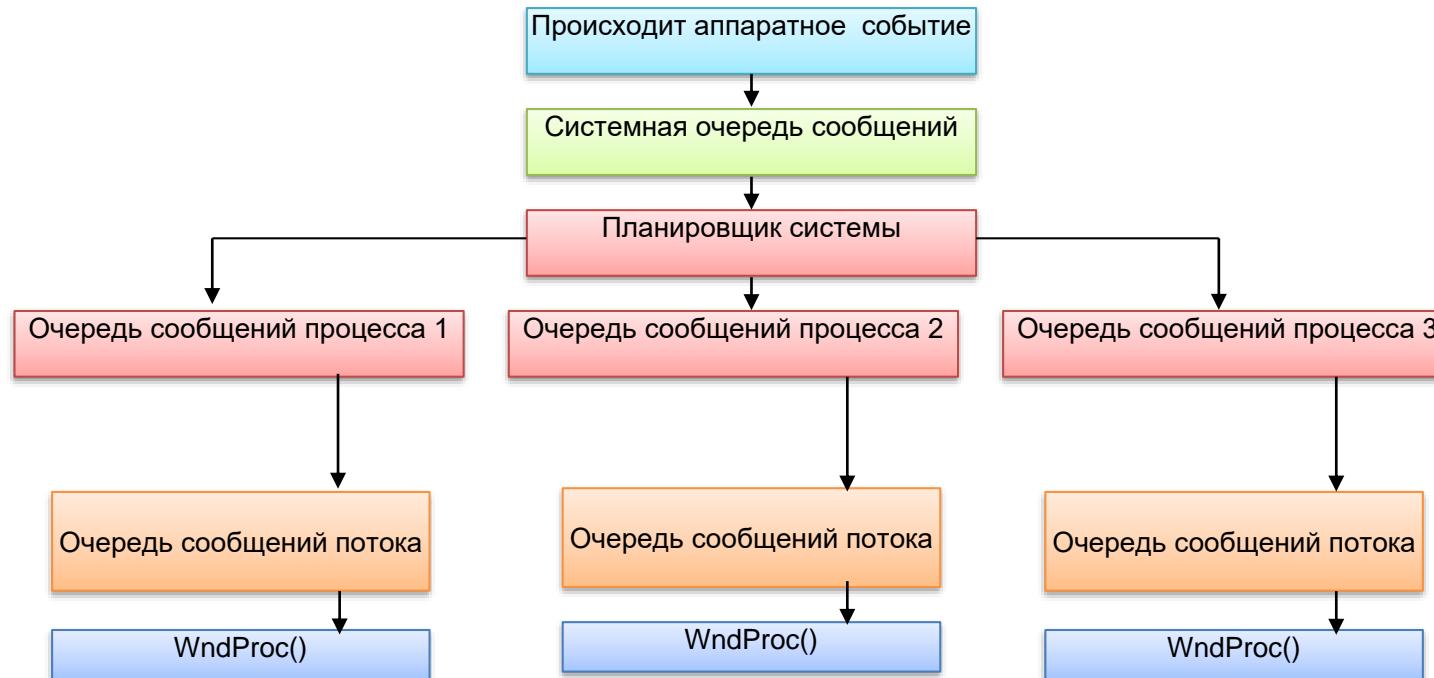
```
HRESULT WndProc(HWND hWnd, UINT message,  
                  WPARAM wParam, LPARAM lParam)  
{  
    switch (message.msg)  
    {  
        case WM_KEYDOWN:  
            //handle  
            break;  
        case WM_LBUTTONDOWN:  
            // handle  
            break;  
        case WM_TIMER:  
            // handle  
            break;  
        case WM_MOVE:  
            // handle  
            break;  
    }  
}
```



MFC, Windows Forms

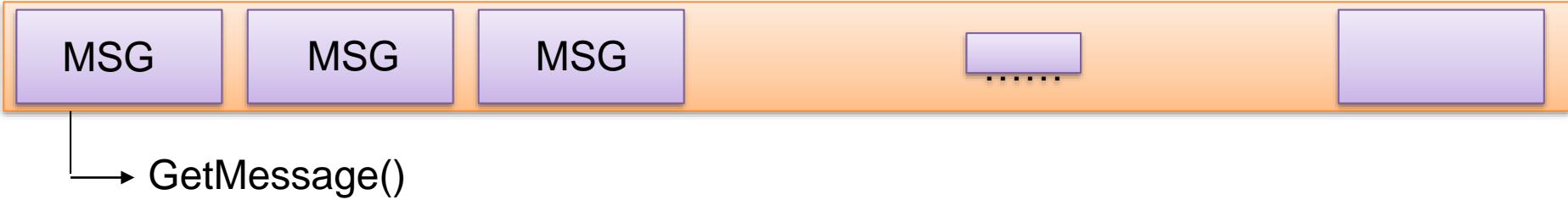
CQG

# Пользовательский интерфейс: Архитектура обработки сообщений



# Пользовательский интерфейс: Очередь сообщений

PostMessage() →



```
void RunMessageLoop() {  
    MSG msg;  
    while (GetMessage(&msg, 0, 0, 0)) {  
        TranslateMessage(&msg);  
        DispatchMessage(&msg);  
    }  
}
```

```
LRESULT WindowProc(HWND hwnd, UINT message,  
                   WPARAM wParam, LPARAM lParam) {  
    switch (message) {  
        case WM_KEYDOWN: /* process keyboard */ break;  
        case WM_LBUTTONDOWN: /* process mouse */ break;  
        case WM_TIMER: /* process timer */ break;  
    }  
}
```

<https://docs.microsoft.com/en-us/windows/win32/learnwin32/your-first-windows-program>

# Пользовательский интерфейс: Функции обработки сообщений

```
BOOL WINAPI GetMessage(  
    LPMSG lpMsg, HWND hWnd,  
    UINT wMsgFilterMin, UINT wMsgFilterMax);
```

- извлекает сообщение из очереди сообщений вызывающего потока
- блокирует выполнение программы, если сообщений нет
- при блокировке не занимает процессорное время

```
BOOL WINAPI TranslateMessage(const MSG *lpMsg);
```

- переводит сообщения формата виртуальных клавиш в сообщения символы

```
HRESULT WINAPI DispatchMessage(const MSG *lpmsg);
```

- пересыпает сообщение оконной процедуре

```
BOOL WINAPI PeekMessage(  
    LPMSG lpMsg, HWND hWnd,  
    UINT wMsgFilterMin, UINT wMsgFilterMax,  
    UINT wRemoveMsg);
```

- извлекает сообщение из очереди сообщений вызывающего потока
- не блокирует выполнение программы

# Пользовательский интерфейс: Функции отправки сообщений

```
BOOL PostMessage(HWND, UINT, WPARAM, LPARAM);
```

- Определяет, каким потоком создано окно-адресат
- Помещает сообщение в очередь потока
- Возвращает управление
- Нет возможности получить результат обработки сообщения



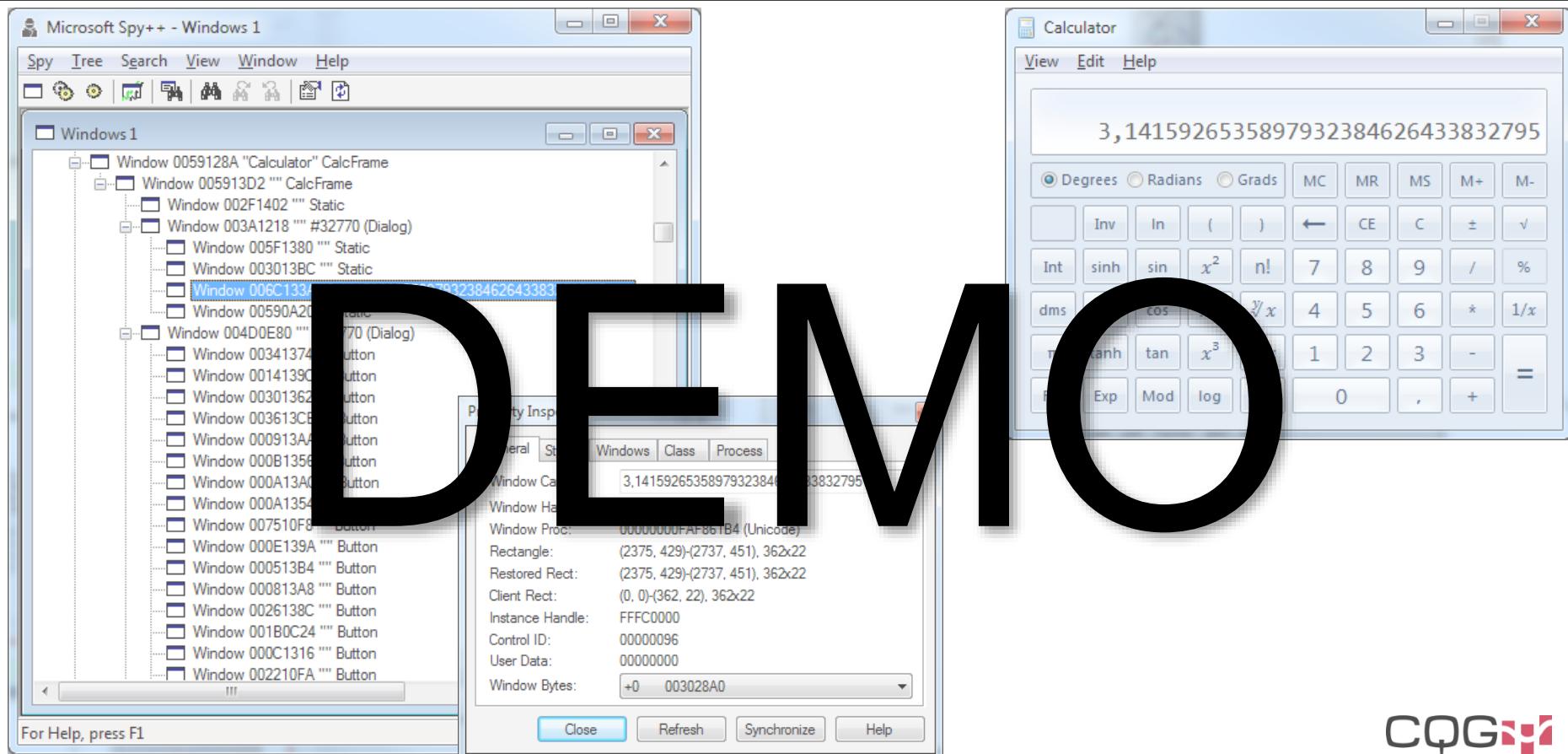
# Пользовательский интерфейс: Функции отправки сообщений

```
LRESULT SendMessage(HWND, UINT, WPARAM, LPARAM);
```

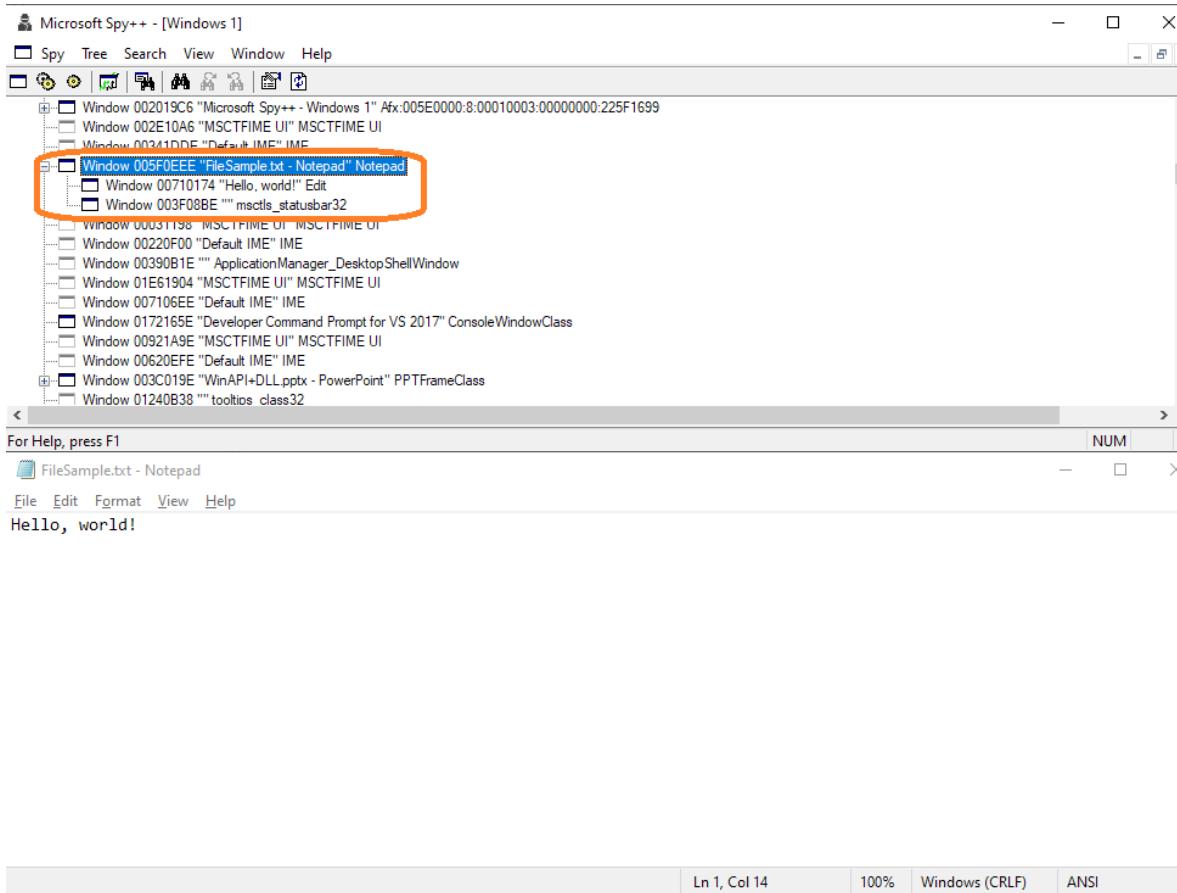
- Блокирует вызывающий поток
- Ожидает, пока окно-адресат не обработает оконное сообщение
- Дает возможность получить результат обработки сообщения



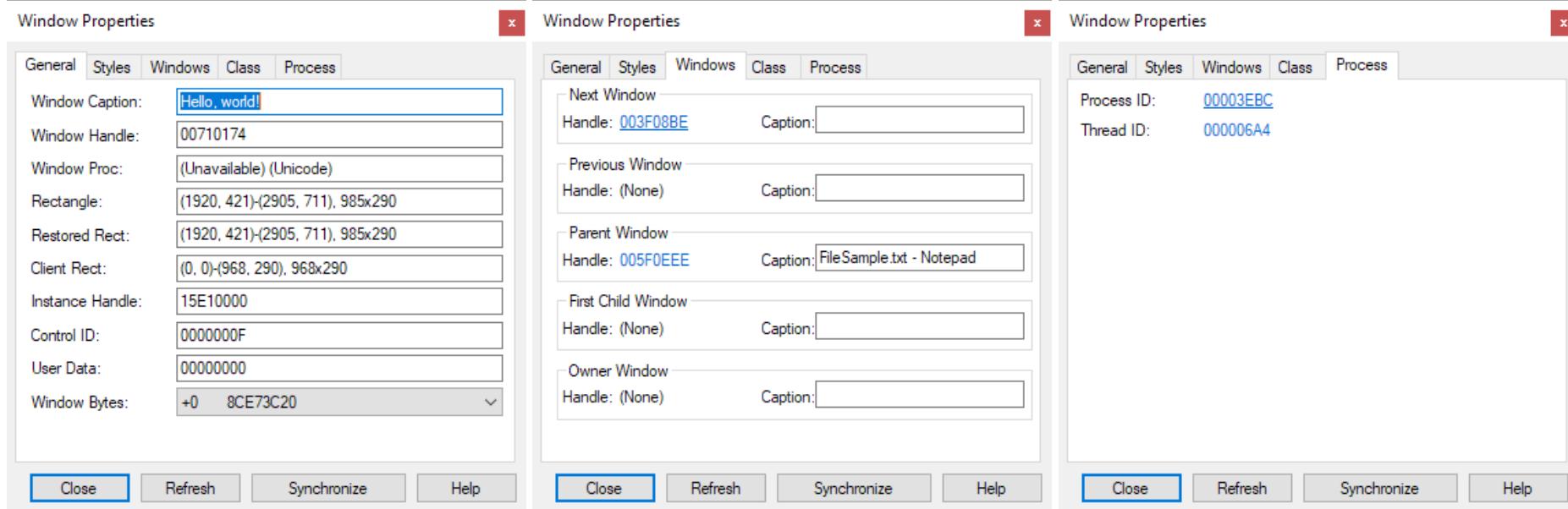
# Пользовательский интерфейс: Окна в Microsoft Spy++



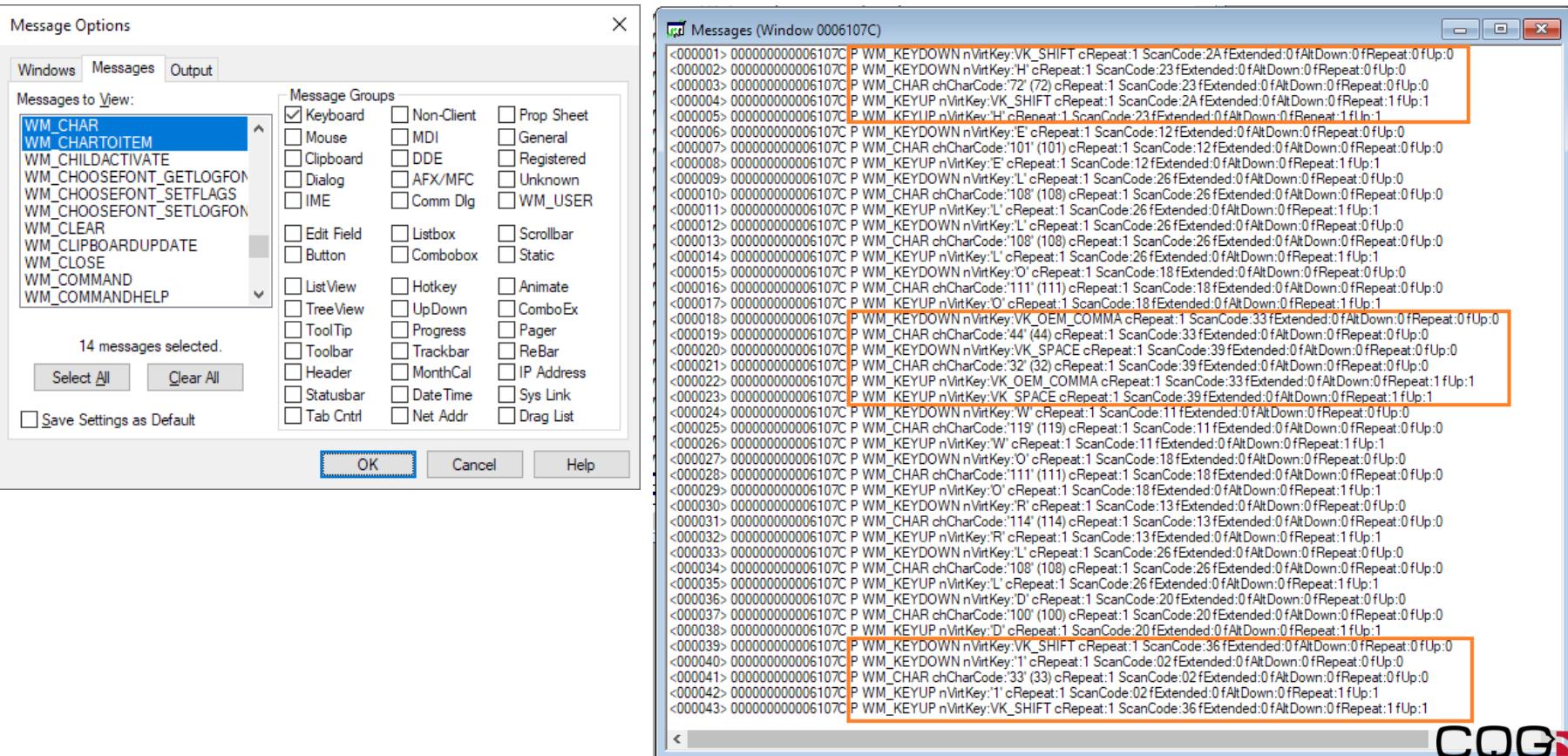
# Пользовательский интерфейс: Окна в Microsoft Spy++



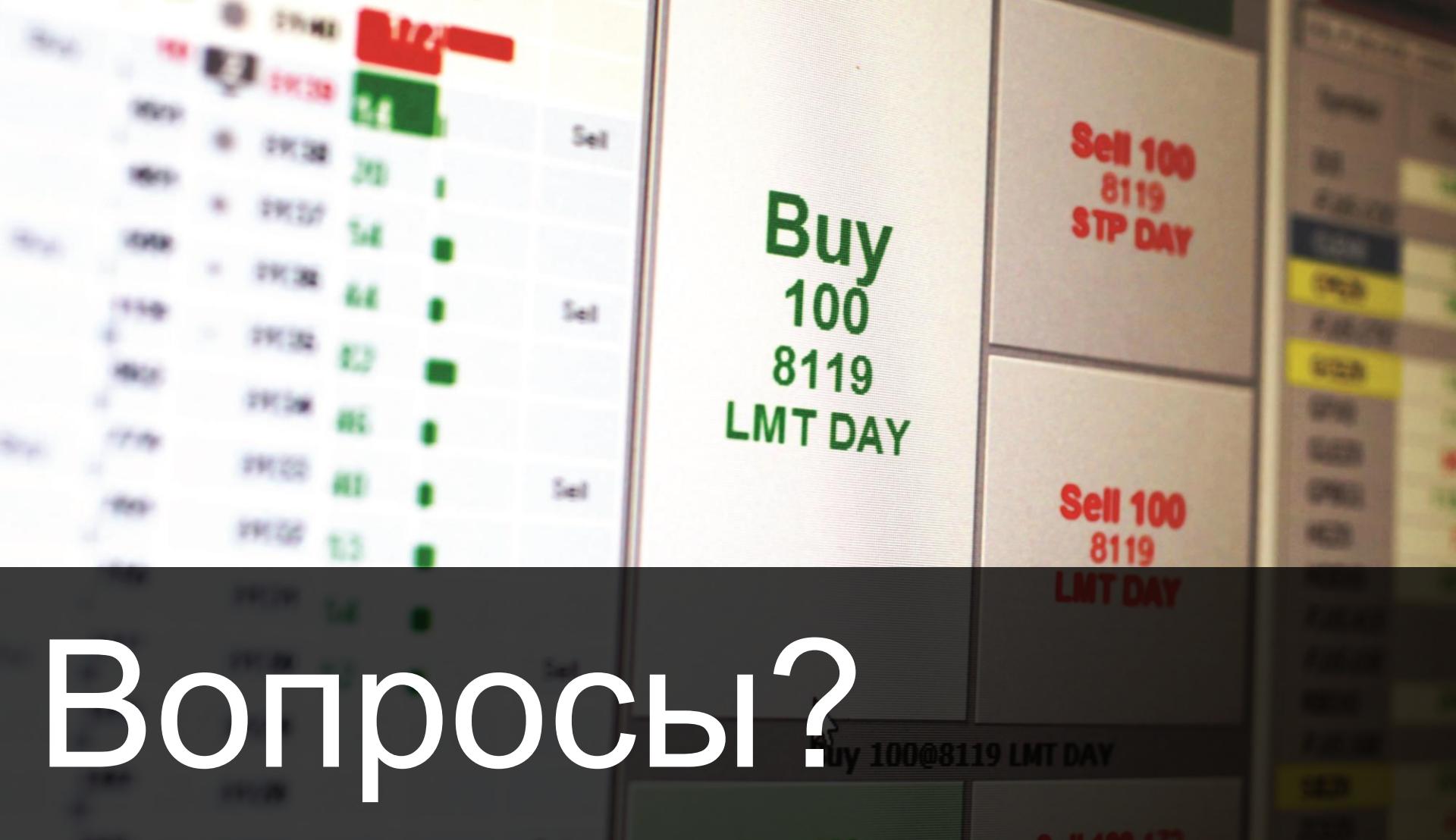
# Пользовательский интерфейс: Окна в Microsoft Spy++



# Пользовательский интерфейс: Окна в Microsoft Spy++



# Вопросы?



# Динамические библиотеки (DLL)

- Общие сведения
  - Предыстория
  - Жизненный цикл разработки
  - Виды использования
  - Цели использования
  - Адресное пространство процесса
  - Экспортируемые функции
  - Функция входа/выхода
- Использование DLL в коде
  - Алгоритм загрузки библиотек
  - Явное связывание
  - Неявное связывание
  - Искажение имен
- Недостатки DLL



# Динамические библиотеки: Предыстория

Dynamic Link Library – концепция совместно используемых библиотек в ОС Windows.

- Изначально предназначено для эффективного управления памятью.
- Впоследствии стало средством для эффективной разработки ПО за счёт модульности.

Пример:

Photoshop.exe

ImageProc.dll

LicenseVerifier.dll

kernel32.dll

ImageEffects.dll

PluginManager.dll

...

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

## Динамически библиотеки (.DLL)

- Разрабатываются независимо

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

## Динамически библиотеки (.DLL)

- Разрабатываются независимо
- Не участвуют в линковке приложения (.EXE модуля)

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

## Динамически библиотеки (.DLL)

- Разрабатываются независимо
- Не участвуют в линковке приложения (.EXE модуля)
- Могут использоваться независимо от .EXE модуля

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

## Динамически библиотеки (.DLL)

- Разрабатываются независимо
- Не участвуют в линковке приложения (.EXE модуля)
- Могут использоваться независимо от .EXE модуля
- Могут поставляться пользователям

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

## Динамически библиотеки (.DLL)

- Разрабатываются независимо
- Не участвуют в линковке приложения (.EXE модуля)
- Могут использоваться независимо от .EXE модуля
- Могут поставляться пользователям
- Могут загружаться и выгружаться динамически

# Динамические библиотеки: Жизненный цикл разработки

## Статические библиотеки (.LIB)

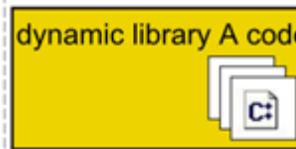
- Разрабатываются независимо
- При линковке приложения (.EXE модуля), включаются в его состав
- Не используются независимо от .EXE модуля
- Не поставляются пользователям
- При изменении одной библиотеки необходимо перелинковывать весь модуль

## Динамически библиотеки (.DLL)

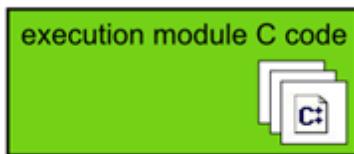
- Разрабатываются независимо
- Не участвуют в линковке приложения (.EXE модуля)
- Могут использоваться независимо от .EXE модуля
- Могут поставляться пользователям
- Могут загружаться и выгружаться динамически
- Поддержка загрузки .DLL лежит на операционной системе

# Динамические библиотеки: Жизненный цикл разработки

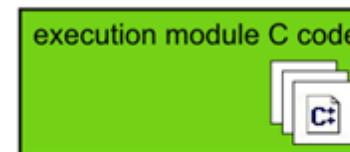
Статические  
библиотеки  
(LIB)



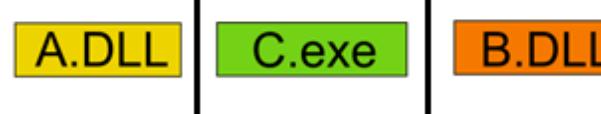
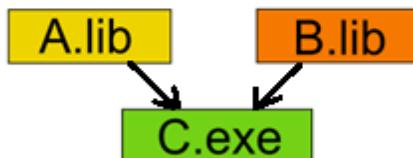
Динамические  
библиотеки  
(DLL)



Разработка



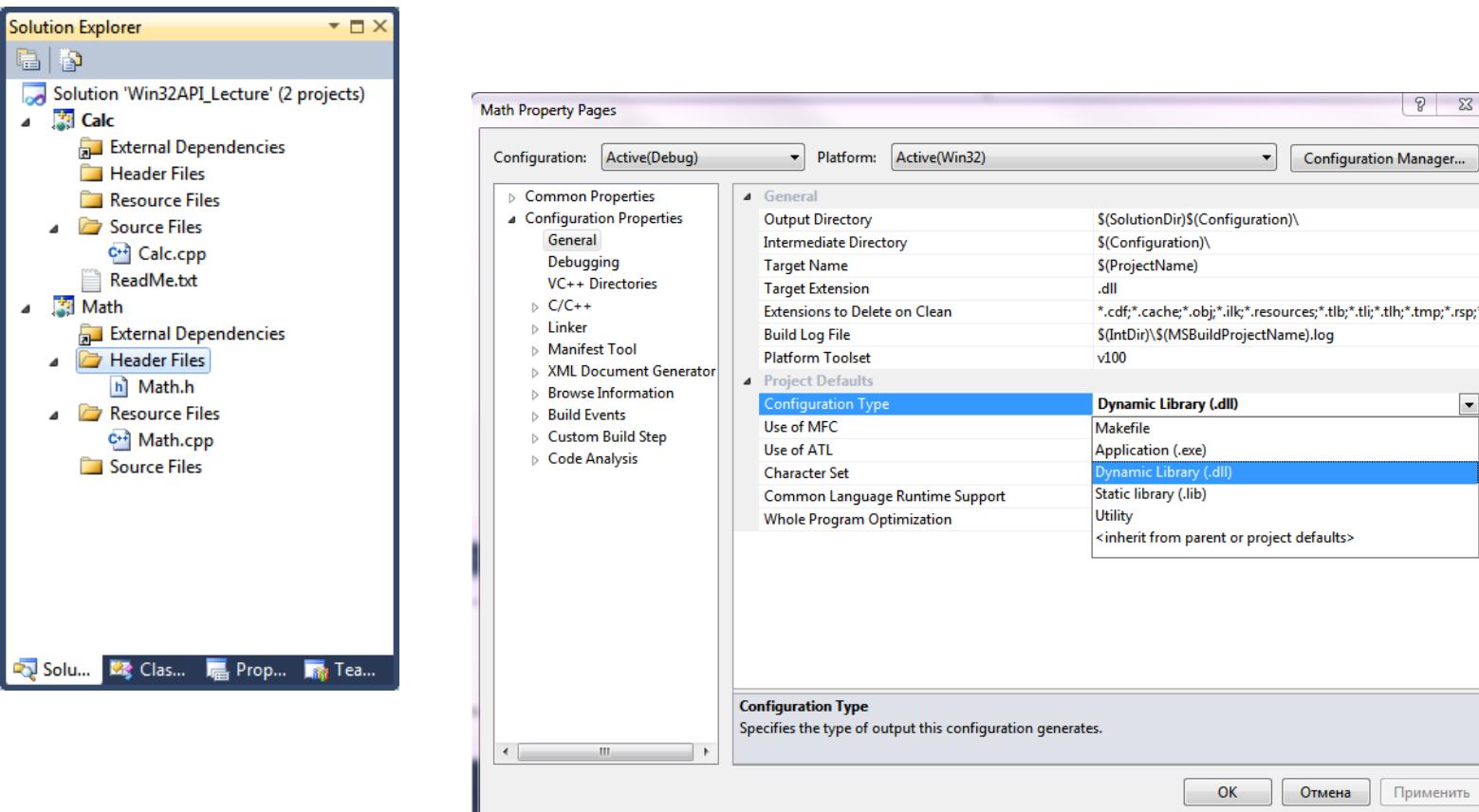
Компоновка



Поставка



# Динамические библиотеки: Типы проектов



# Динамические библиотеки: Виды использования



Динамическая загрузка набора библиотек

Использование одной библиотеки разными приложениями



# Динамические библиотеки: Цели использования

## Раздельное использование

- Уменьшение использования дискового пространства за счет раздельного использования библиотек

# Динамические библиотеки: Цели использования

## Раздельное использование

- Уменьшение использования дискового пространства за счет раздельного использования библиотек
- Уменьшение использования памяти

# Динамические библиотеки: Цели использования

## Раздельное использование

- Уменьшение использования дискового пространства за счет раздельного использования библиотек
- Уменьшение использования памяти

## Динамическая загрузка

- Возможность загрузки и выгрузки компонентов *на лету*

# Динамические библиотеки: Цели использования

## Раздельное использование

- Уменьшение использования дискового пространства за счет раздельного использования библиотек
- Уменьшение использования памяти

## Динамическая загрузка

- Возможность загрузки и выгрузки компонентов *на лету*
- Возможность дополнения программы плагинами

# Динамические библиотеки: Цели использования

## Раздельное использование

- Уменьшение использования дискового пространства за счет раздельного использования библиотек
- Уменьшение использования памяти

## Динамическая загрузка

- Возможность загрузки и выгрузки компонентов *на лету*
- Возможность дополнения программы плагинами

## Упрощение поддержки программы

- Программа разделена на независимые компоненты

# Динамические библиотеки: Цели использования

## Раздельное использование

- Уменьшение использования дискового пространства за счет раздельного использования библиотек
- Уменьшение использования памяти

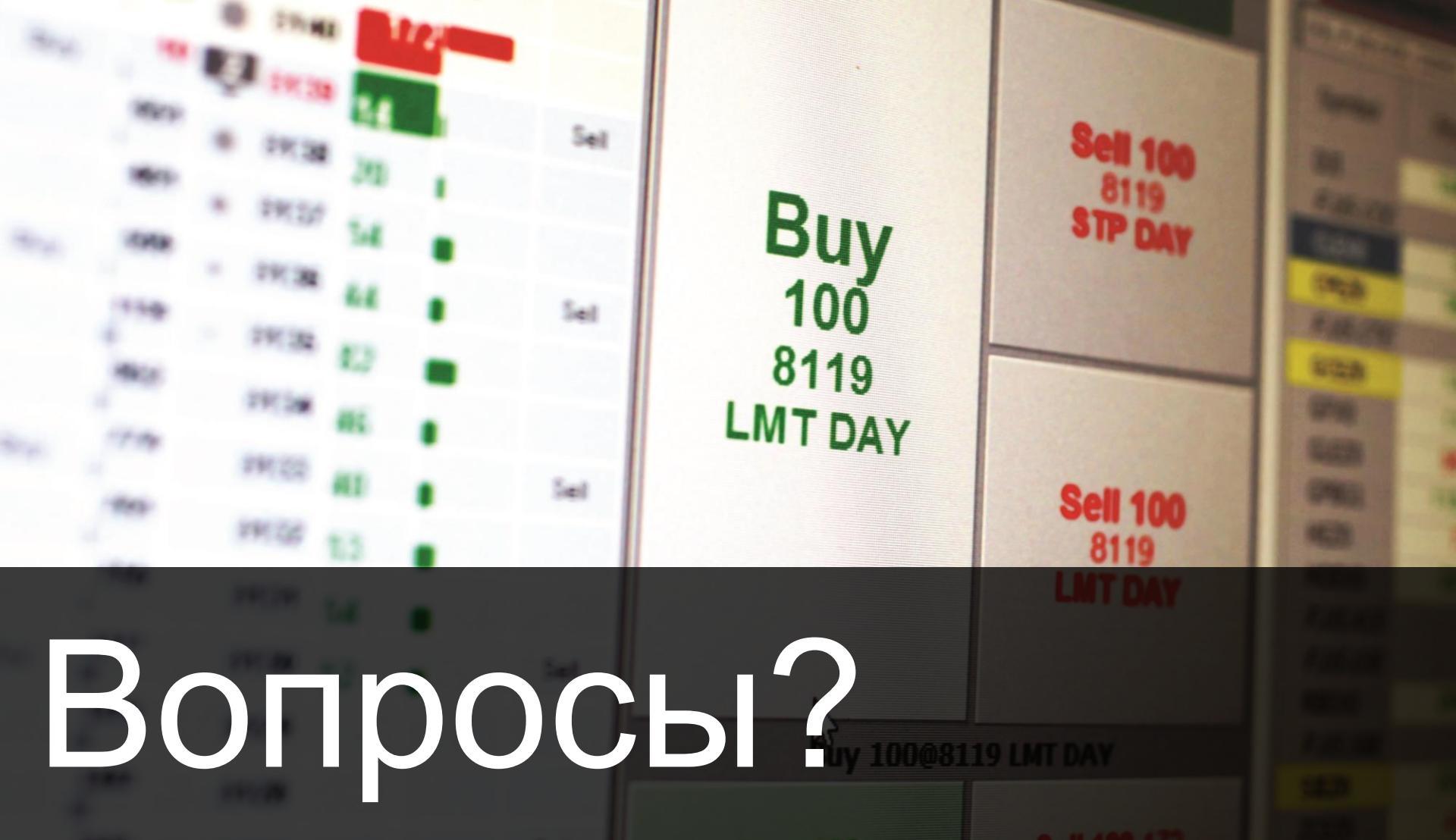
## Динамическая загрузка

- Возможность загрузки и выгрузки компонентов *на лету*
- Возможность дополнения программы плагинами

## Упрощение поддержки программы

- Программа разделена на независимые компоненты
- Исправление багов не требует полной переустановки программы

# Вопросы?



# Динамические библиотеки: Адресное пространство

## 1. Создание процесса и выделение виртуальной памяти



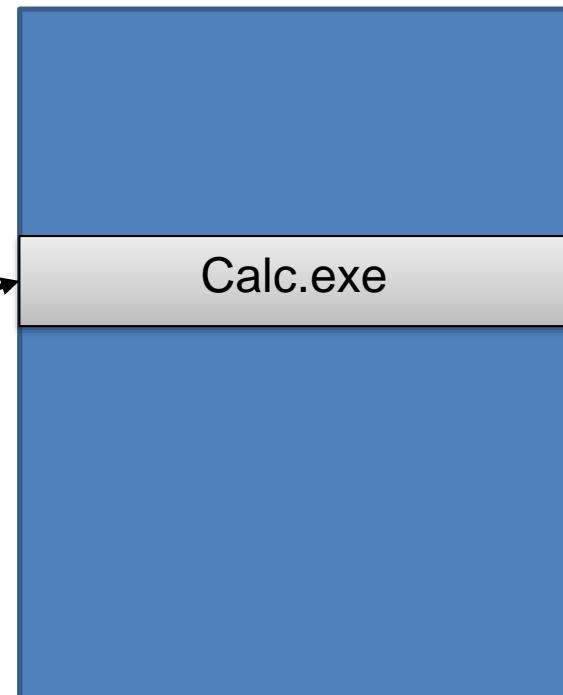
# Динамические библиотеки: Адресное пространство

1. Создание процесса и выделение памяти
2. Загрузка EXE в память

0x7FFF'FFFF

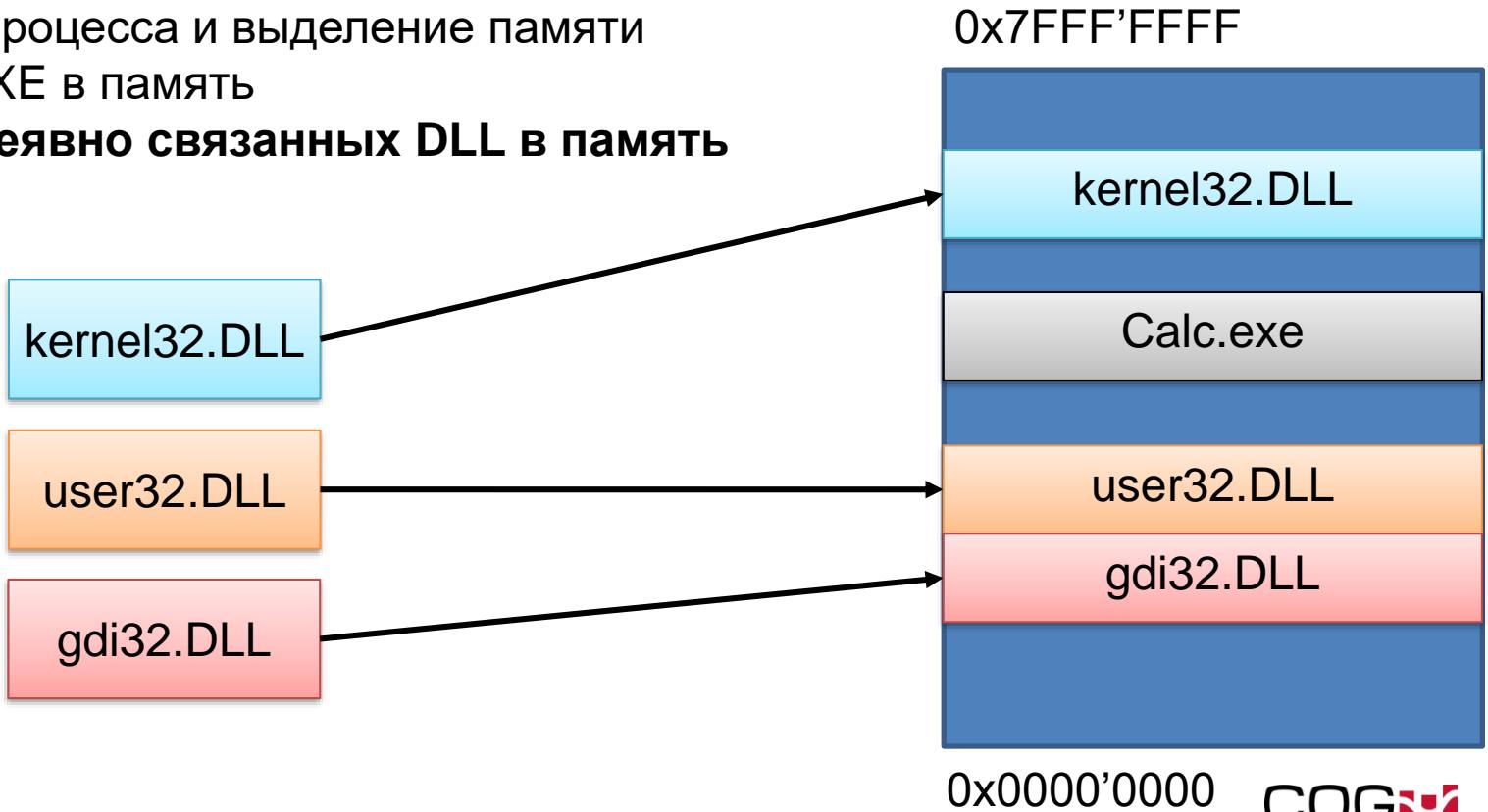
Calc.EXE

```
void _cdecl func() {
00411A20 55          push    ebp
00411A21 8B EC        mov     ebp,esp
00411A23 81 EC C0 00 00 00 sub    esp,0C0h
00411A29 53          push    ebx
00411A2A 56          push    esi
00411A2B 57          push    edi
00411A2C 8D BD 40 FF FF FF lea    edi,[ebp-0C0h]
00411A32 B9 30 00 00 00 mov     ecx,30h
00411A37 B8 CC CC CC CC mov     eax,0CCCCCCCCh
00411A3C F3 AB        rep stos dword ptr [edi]
}
00411A3E 5F          pop     edi
00411A3F 5E          pop     esi
00411A40 5B          pop     ebx
00411A41 8B E5        mov     esp,ebp
00411A43 5D          pop     ebp
00411A44 C3          ret
```

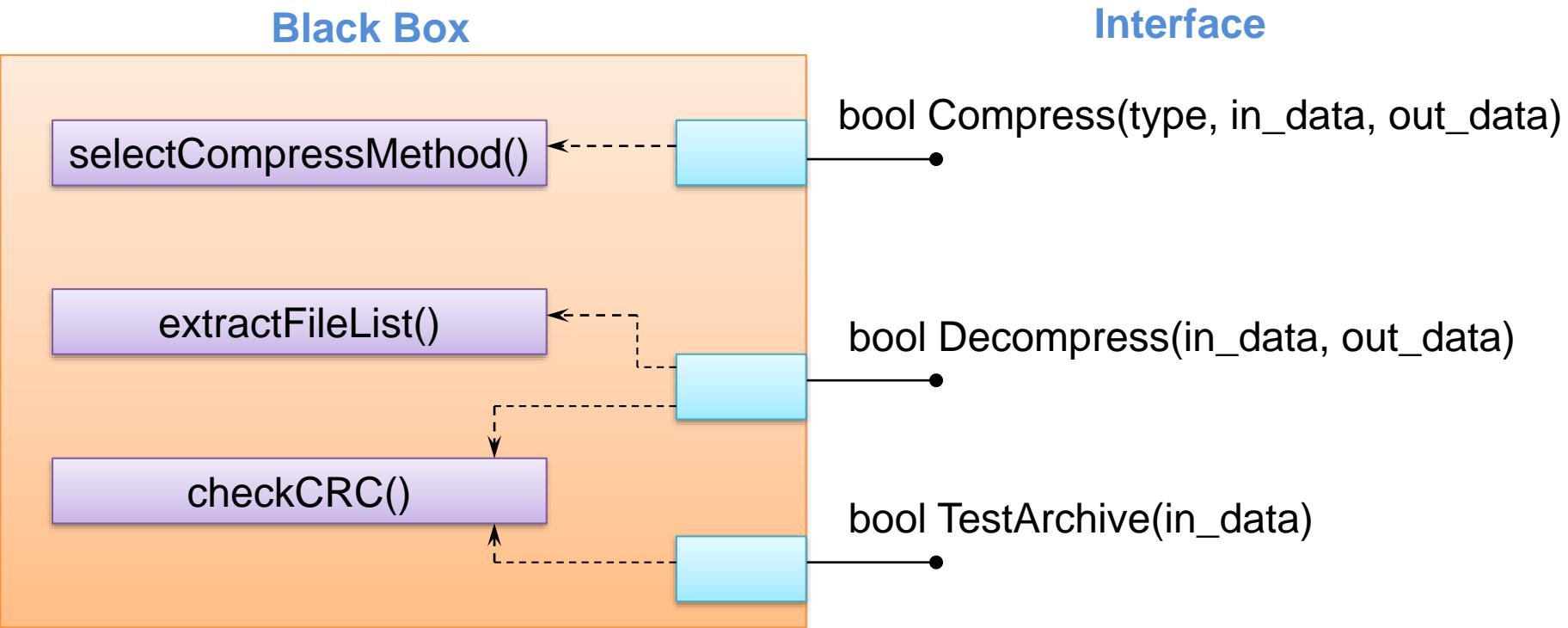


# Динамические библиотеки: Адресное пространство

1. Создание процесса и выделение памяти
2. Загрузка EXE в память
3. **Загрузка неявно связанных DLL в память**



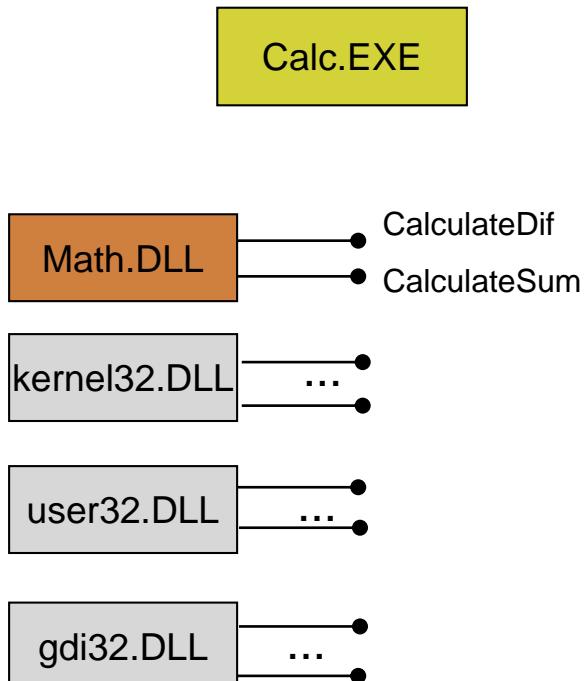
# Динамические библиотеки: Экспортируемые функции



MSDN: Implementation changes. Interfaces don't.

# Динамические библиотеки: Экспортируемые функции

Адресное пространство процесса



D:\Projects\Exportimport>**dumpbin -exports Math.dll**  
Microsoft (R) COFF/PE Dumper Version 7.10.6030  
Copyright (C) Microsoft Corporation. All rights reserved.

Dump of file Math.dll

File Type: DLL

Section contains the following exports for Math.dll

00000000 characteristics  
494018C6 time date stamp Wed Dec 10 22:30:14 2008  
0.00 version  
1 ordinal base  
2 number of functions  
2 number of names

ordinal	hint	RVA	name
1	0	0001C514	CalculateDif
2	1	0001C5BE	CalculateSum

Summary

5000 .data  
1000 .idata  
7000 .rdata  
3000 .reloc  
3B000 .text  
1B000 .textbss

# Динамические библиотеки: Экспортируемые функции

## Способы определения экспортируемых функций

- `_declspec(dllexport)`

```
extern "C"  
_declspec(dllexport) int CalculateSum(int a, int b)  
{  
    return a + b;  
}
```

> link.exe /DLL Math.obj

- `/EXPORT`

```
extern "C" int CalculateSum(int a, int b)  
{  
    return a + b;  
}
```

> link.exe /DLL /EXPORT:CalculateSum Math.obj

- Module definition file (.DEF)
- `#pragma comment(linker, "/export:CalculateSum")`

# Динамические библиотеки: Опциональная точка входа в DLL

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH: загрузка DLL в адресное пространство процесса
            break;
        case DLL_THREAD_ATTACH: создаётся поток
            break;
        case DLL_THREAD_DETACH: поток корректно завершается
            break;
        case DLL_PROCESS_DETACH: выгрузка DLL из адресного пространства процесса
            break;
    }
    return TRUE;
}
```

# Динамические библиотеки: Способы задания точки входа в DLL

- Ключи /NOENTRY и /ENTRY не указаны. DLL имеет точку входа по умолчанию. Линковщик MS Visual Studio установит ее равной \_DIIIMainCRTStartup. В конце выполнения \_DIIIMainCRTStartup вызовет функцию DIIIMain, если она определена.

```
> link.exe /DLL Math.obj
```

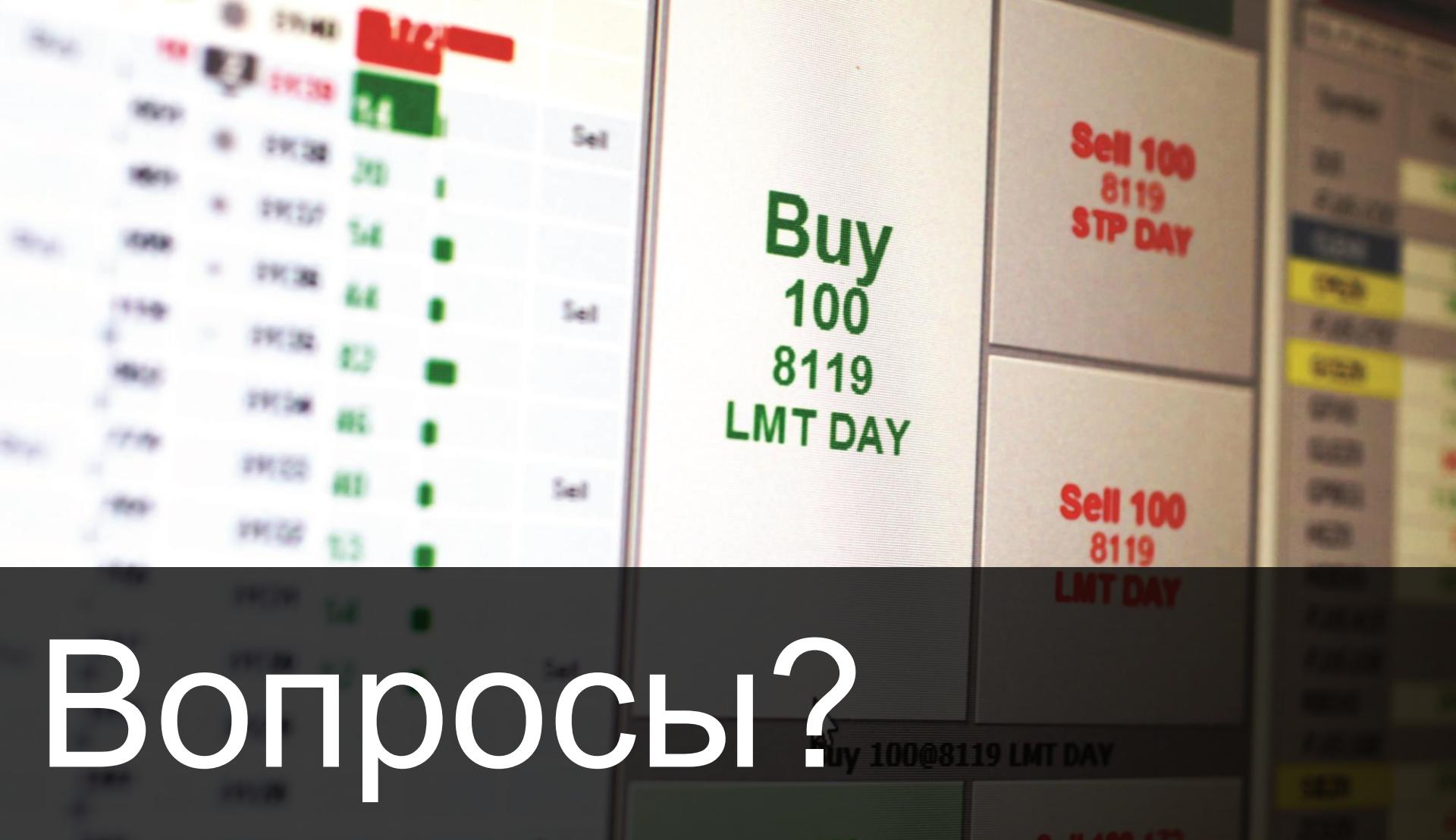
- Указан ключ /NOENTRY линковщика. Точка входа отсутствует.

```
> link.exe /DLL /NOENTRY Math.obj
```

- Указан ключ /ENTRY линковщика.

```
> link.exe /DLL /ENTRY:MyDIIIMain Math.obj
```

# Вопросы?





# Динамические библиотеки (DLL)

Использование DLL в коде



# Динамические библиотеки: Явное связывание

calc.cpp

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

typedef int (*CalcFuncPtr)(int a, int b);

int main() {
    HMODULE h = ::LoadLibrary("math.dll");

    CalcFuncPtr calcSum =
        (CalcFuncPtr) ::GetProcAddress(h, "CalculateSum");
    CalcFuncPtr calcDiff =
        (CalcFuncPtr) ::GetProcAddress(h, "CalculateDif");

    std::cout << calcSum(10, 15) << std::endl;
    std::cout << calcDiff(25, 12) << std::endl;

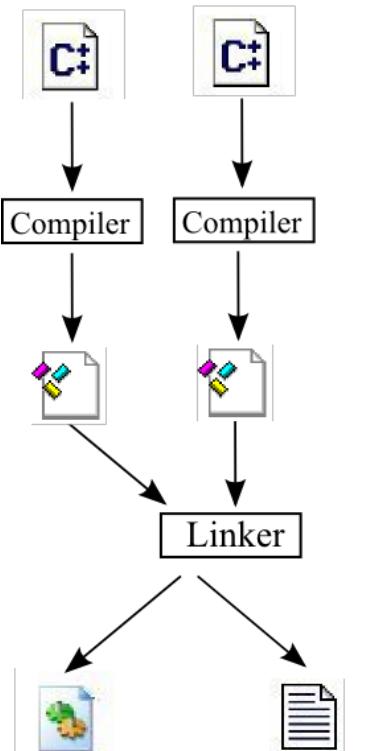
    ::FreeLibrary(h);
    return 0;
}
```

math.cpp

```
extern "C"
__declspec(dllexport) int CalculateSum(int a, int
b)
{
    return a + b;
}

extern "C"
__declspec(dllexport) int CalculateDif(int a, int b)
{
    return a - b;
}
```

# Динамические библиотеки: Явное связывание



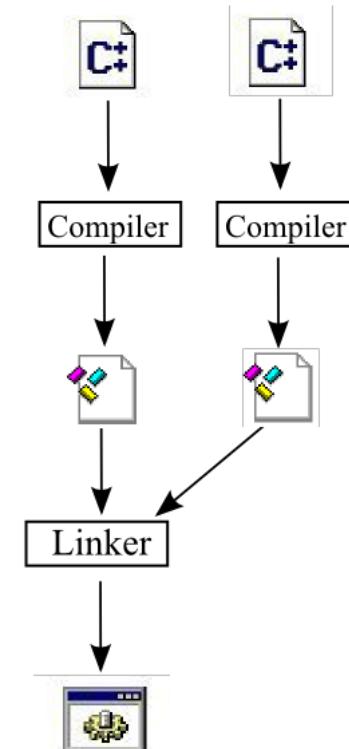
DLL import library (.lib)

```
D:\Projects\ExportImport>dumpbin -imports Calc.exe
Microsoft (R) COFF/PE Dumper Version 7.10.6030
Copyright (C) Microsoft Corporation. All rights reserved.
```

Dump of file Calc.exe

File Type: EXECUTABLE IMAGE

Section contains the following imports:

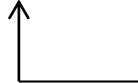


EXE

# Динамические библиотеки: Явное связывание: API

**Загрузка:**

```
HINSTANCE LoadLibrary(PCTSTR pszDIIPathName);
```



Адрес виртуальной памяти,  
по которому спроектирован образ dll файла

**Выгрузка:**

```
BOOL FreeLibrary(HINSTANCE hinstDII);
```

```
HINSTANCE GetModuleHandle(PCTSTR pszModuleHandle);
```

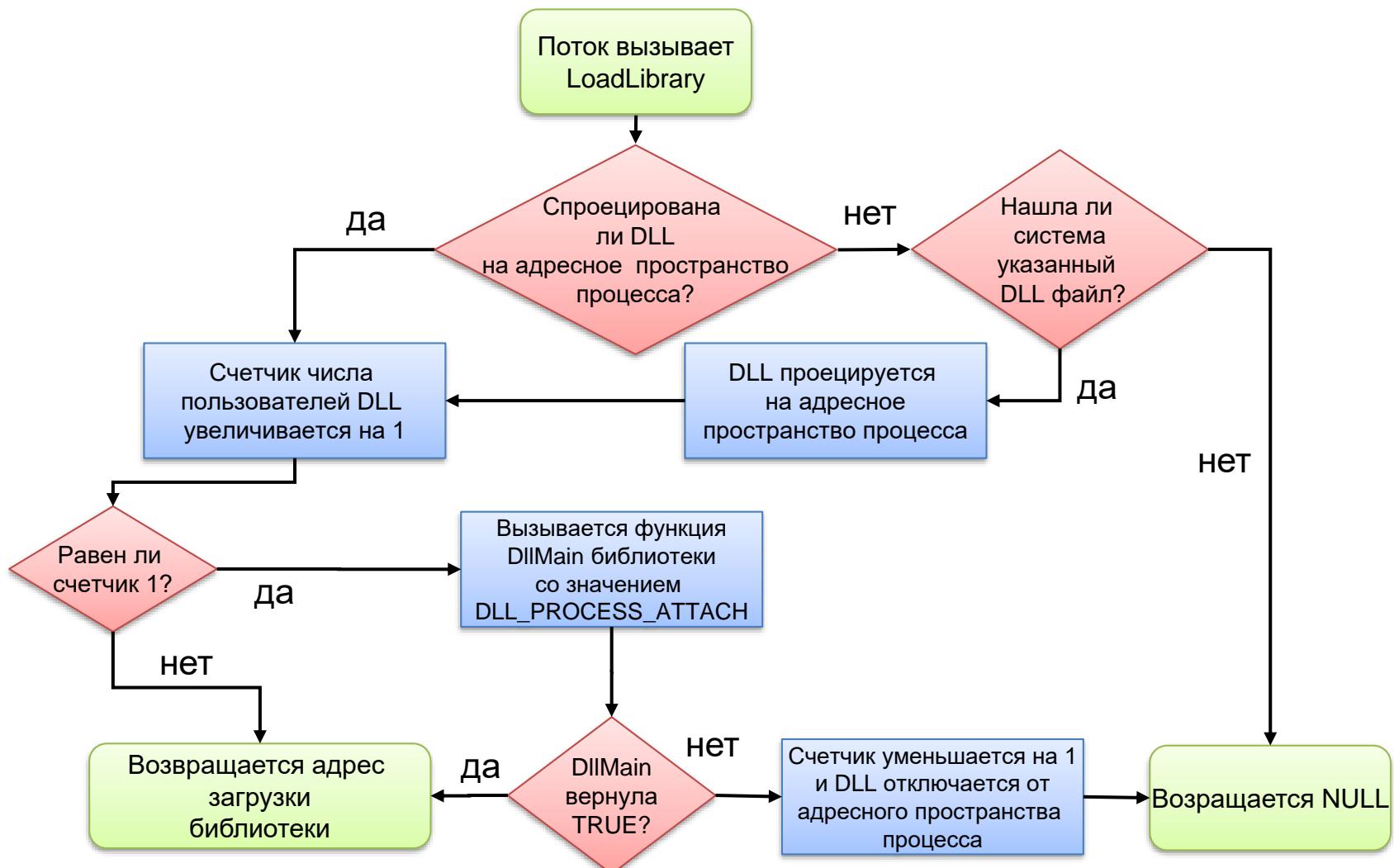
**Адрес экспортируемой функции:**

```
FARPROC GetProcAddress(HINSTANCE hinstDII, PCSTR pszSymbolName);
```

ANSI – имя функции  
Порядковый номер (Ординал)



# Динамические библиотеки: Алгоритм ::LoadLibrary



# Динамические библиотеки: Алгоритм ::LoadLibrary (поиск dll)

## Поиск DLL:

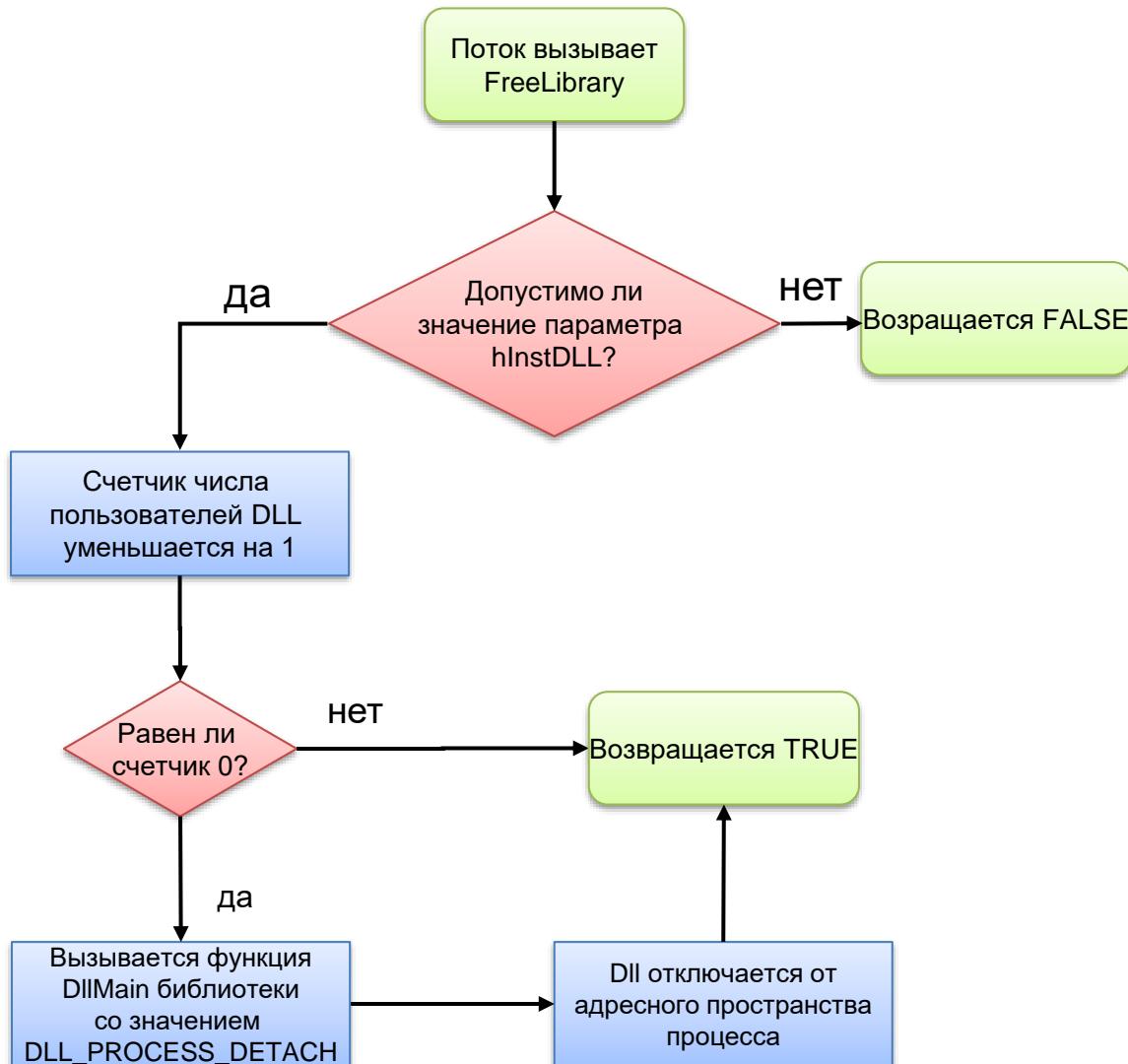
1. Если DLL с таким именем уже загружена, поиск не выполняется
2. Если это Known DLL, то загрузить из C:\Windows\System32
3. Папка, содержащая EXE-файл
4. Текущая папка, если выключен режим **SafeDllSearchMode**
5. Системная папка: C:\Windows\System32 или C:\Windows\SysWOW64
6. 16-bit системная папка: C:\Windows\System
7. Папка Windows: C:\Windows
8. Текущая папка, если включен режим **SafeDllSearchMode** (по умолчанию)
9. Папки из переменной окружения PATH

## Known DLLs:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs

<https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order>

# Динамические библиотеки: Алгоритм ::FreeLibrary



# Динамические библиотеки: Неявное связывание

math.cpp

```
#define MATHAPI extern "C" __declspec(dllexport)  
#include "math.h"  
  
int CalculateSum(int a, int b)  
{  
    return a + b;  
}  
  
int CalculateDif(int a, int b)  
{  
    return a - b;  
}
```

math.h

```
#ifndef MATHAPI  
#define MATHAPI extern "C" __declspec(dllimport)  
#endif  
  
MATHAPI int CalculateSum(int a, int b);  
MATHAPI int CalculateDif(int a, int b);
```

# Динамические библиотеки: Неявное связывание

calc.cpp

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

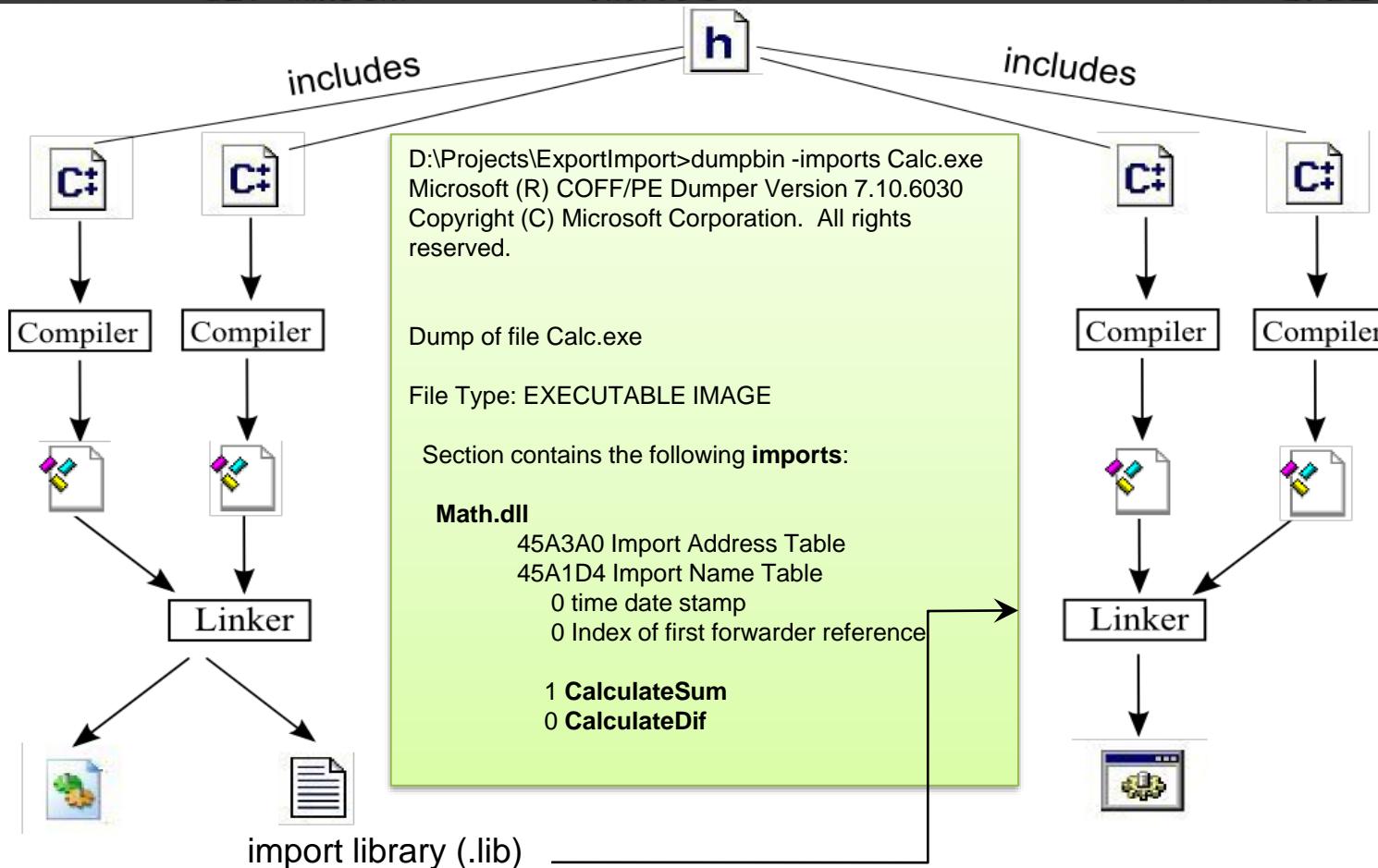
int main()
{
    std::cout << CalculateSum(10, 15) << std::endl;
    std::cout << CalculateDif(25, 12) << std::endl;
    return 0;
}
```

math.h

```
#ifndef MATHAPI
#define MATHAPI extern "C" __declspec(dllexport)
#endif
```

```
MATHAPI int CalculateSum(int a, int b);
MATHAPI int CalculateDif(int a, int b);
```

# Динамические библиотеки: Неявное связывание



# Динамические библиотеки: Библиотека импорта

> link.exe /DLL Math.obj

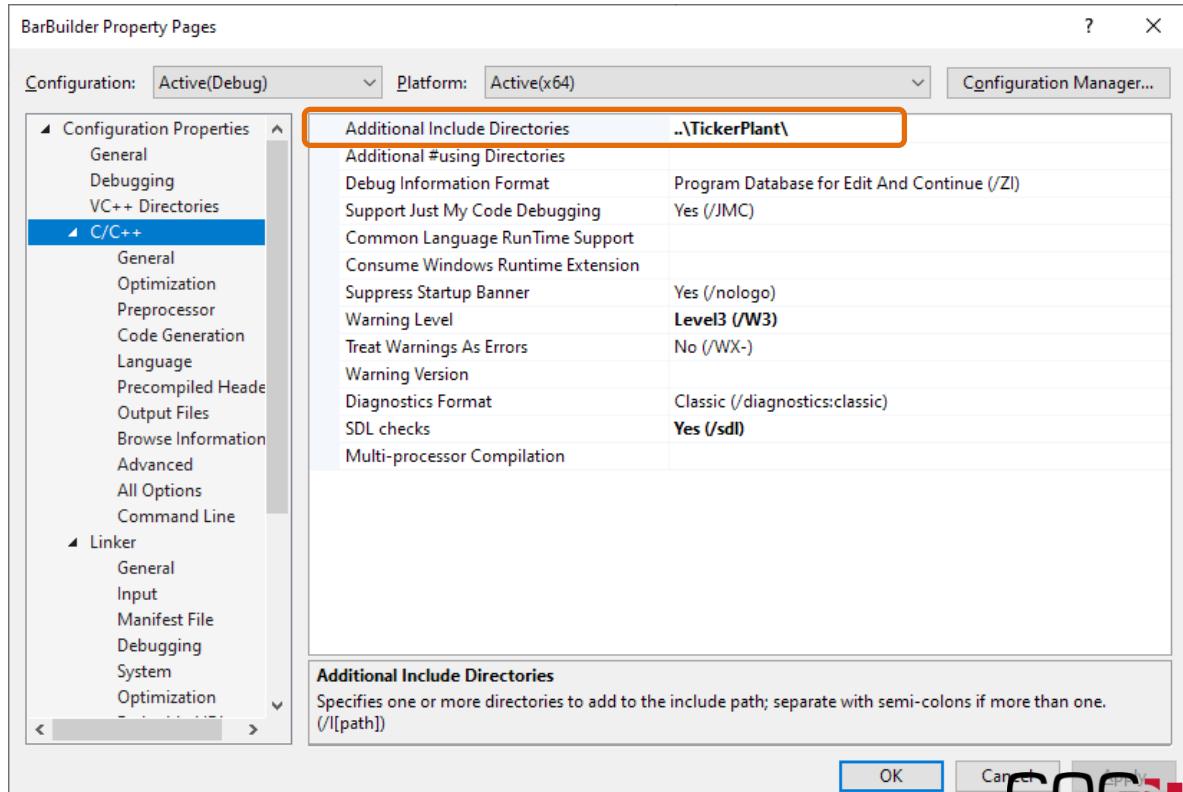
Math.dll

Math.lib

- Math.lib не содержит код
- Math.lib содержит имена экспортруемых из Math.dll функций
- Создается всегда при линковке динамических библиотек
- Не имеет смысла отдельно от Math.dll
- Компоновщик EXE модуля использует ее при **неявном связывании**
- Компоновщик EXE модуля **не** использует ее при **явном связывании**
- *Не путать со статическими библиотеками! (у них то же расширение)*

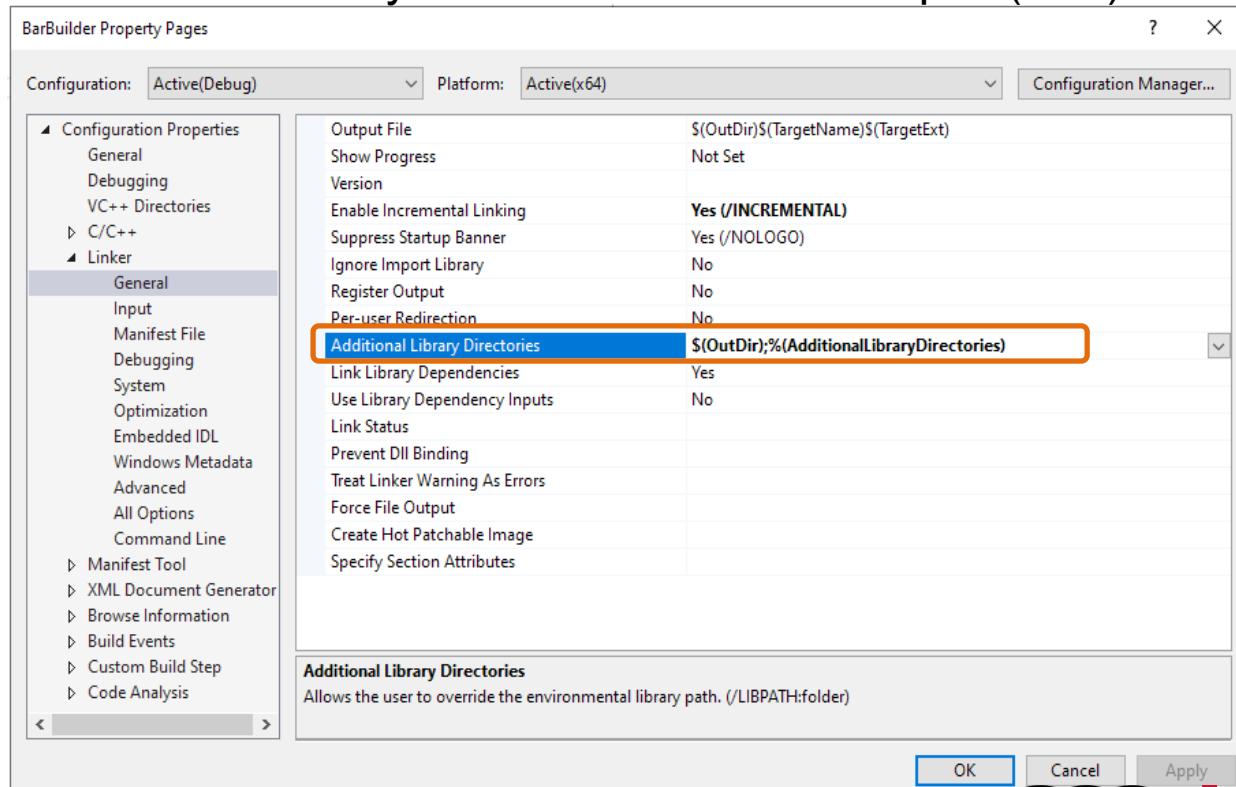
# Динамические библиотеки: Неявное связывание (настройка)

**Additional Include Directories** обозначает путь к заголовочным файлам библиотек для неявного связывания



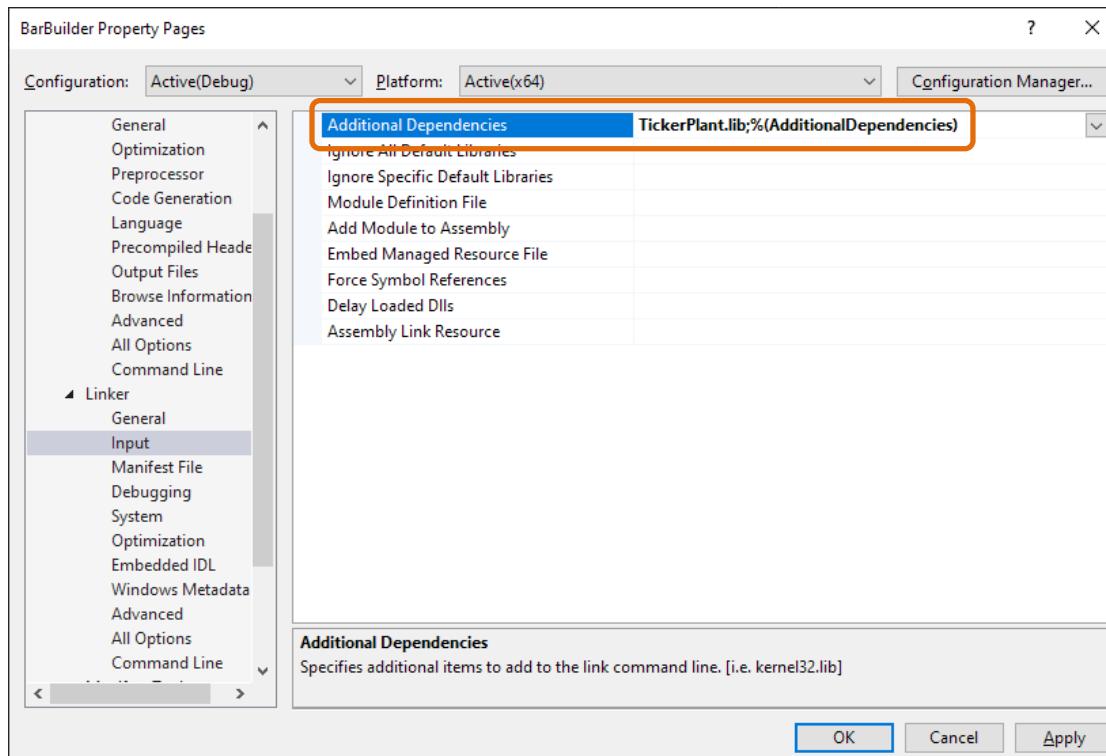
# Динамические библиотеки: Неявное связывание (настройка)

**Additional Library Directories** обозначает путь к библиотекам импорта (.LIB)  
для неявного связывания



# Динамические библиотеки: Неявное связывание (настройка)

В Additional Dependencies перечислены библиотеки импорта (.LIB) для неявного связывания



# Динамические библиотеки: Виды связывания (итог)

С точки зрения библиотеки

- Библиотека содержит экспортные функции
- Линковщик создает библиотеку DLL и библиотеку импорта для нее

# Динамические библиотеки: Виды связывания (итог)

С точки зрения исполняемого модуля

## Явное связывание

- Не требуется конфигурация линковщика исполняемого EXE модуля, работу выполняет ОС
- Код EXE модуля должен содержать вызовы загрузки/выгрузки DLL
- Загрузка и выгрузка библиотеки происходит по запросу
- Таблица импорта EXE пуста

# Динамические библиотеки: Виды связывания (итог)

С точки зрения исполняемого модуля

## Явное связывание

- Не требуется конфигурация линковщика исполняемого EXE модуля, работу выполняет ОС
- Код EXE модуля должен содержать вызовы загрузки/выгрузки DLL
- Загрузка и выгрузка библиотеки происходит по запросу
- Таблица импорта EXE пуста

## Неявное связывание

- Требуется конфигурация линковщика исполняемого EXE модуля
- Код EXE модуля должен включать общий заголовочный файл с определением импортируемых функций
- Загрузка библиотеки происходит в начале работы программы (к примеру, системные DLL) по таблице импорта
- Таблица импорта EXE **непуста**

# Динамические библиотеки: Искажение имён (name mangling)

```
#ifndef MATHAPI  
#define MATHAPI __declspec(dllexport)  
#endif  
  
MATHAPI int CalculateSum(int a, int b)
```

```
#define MATHAPI __declspec(dllexport)  
#include "math.h"  
  
int CalculateSum(int a, int b) {  
    return a + b;  
}
```

```
#ifndef MATHAPI  
#define MATHAPI extern “C” __declspec(dllexport)  
#endif  
  
MATHAPI int CalculateSum(int a, int b)
```

```
#define MATHAPI extern “C” __declspec(dllexport)  
#include "math.h"  
  
int CalculateSum(int a, int b) {  
    return a + b;  
}
```

```
D:\Projects\Exportimport>dumpbin -exports Math.dll  
Microsoft (R) COFF/PE Dumper Version 7.10.6030  
Copyright (C) Microsoft Corporation. All rights reserved.
```

Dump of file Math.dll

File Type: DLL

Section contains the following exports for Math.dll

```
ordinal hint RVA      name  
1 0 00011267 ?CalculateSum@@YAHPAUSOMESTRUCT@@@Z
```

```
D:\Projects\Exportimport>dumpbin -exports Math.dll  
Microsoft (R) COFF/PE Dumper Version 7.10.6030  
Copyright (C) Microsoft Corporation. All rights reserved.
```

Dump of file Math.dll

File Type: DLL

Section contains the following exports for Math.dll

```
ordinal hint RVA      name  
1 0 00011267 CalculateSum
```

# Динамические библиотеки: Искажение имён (name mangling)

C++ компилятор вынужден искажать функции из-за перегрузки функций в языке

# Динамические библиотеки: Искажение имён (name mangling)

C++ компилятор вынужден искажать функции из-за перегрузки функций в языке

**Когда искажение не имеет значения:**

- При разработке in house библиотек с неявным связыванием.  
Получится так:

```
D:\ImplicitLinkage>dumpbin /imports calc.exe | grep Calculate
0 ?CalculateDif@@YAHHH@Z
1 ?CalculateSum@@YAHHH@Z
```

```
D:\ImplicitLinkage>dumpbin /exports math.dll | grep Calculate
1 0 00001010 ?CalculateDif@@YAHHH@Z
2 1 00001000 ?CalculateSum@@YAHHH@Z
```

# Динамические библиотеки: Искажение имён (name mangling)

C++ компилятор вынужден искажать функции из-за перегрузки функций в языке

**Когда искажение не имеет значения:**

- При разработке in house библиотек с неявным связыванием.  
Получится так:

```
D:\ImplicitLinkage>dumpbin /imports calc.exe | grep Calculate
0 ?CalculateDif@@YAHHH@Z
1 ?CalculateSum@@YAHHH@Z
```

```
D:\ImplicitLinkage>dumpbin /exports math.dll | grep Calculate
1 0 00001010 ?CalculateDif@@YAHHH@Z
2 1 00001000 ?CalculateSum@@YAHHH@Z
```

**Когда искажение имеет значение, но с ним можно жить:**

- При разработке in house библиотек с явным связыванием

# Динамические библиотеки: Искажение имён (name mangling)

C++ компилятор вынужден искажать функции из-за перегрузки функций в языке

## Когда искажение не имеет значения:

- При разработке in house библиотек с неявным связыванием.  
Получится так:

```
D:\ImplicitLinkage>dumpbin /imports calc.exe | grep Calculate
0 ?CalculateDif@@YAHNN@Z
1 ?CalculateSum@@YAHNN@Z
```

```
D:\ImplicitLinkage>dumpbin /exports math.dll | grep Calculate
1 0 00001010 ?CalculateDif@@YAHNN@Z
2 1 00001000 ?CalculateSum@@YAHNN@Z
```

## Когда искажение имеет значение, но с ним можно жить:

- При разработке in house библиотек с явным связыванием

## Когда искажение является препятствием:

- При необходимости совмещать разные версии компиляторов
- При необходимости подключать библиотеку на языке С

# Динамические библиотеки: Искажение имён (name mangling)

C++ компилятор вынужден искажать функции из-за перегрузки функций в языке

## Когда искажение не имеет значения:

- При разработке in house библиотек с неявным связыванием.  
Получится так:

```
D:\ImplicitLinkage>dumpbin /imports calc.exe | grep Calculate
0 ?CalculateDif@@YAHNN@Z
1 ?CalculateSum@@YAHNN@Z
```

```
D:\ImplicitLinkage>dumpbin /exports math.dll | grep Calculate
1 0 00001010 ?CalculateDif@@YAHNN@Z
2 1 00001000 ?CalculateSum@@YAHNN@Z
```

## Когда искажение имеет значение, но с ним можно жить:

- При разработке in house библиотек с явным связыванием

## Когда искажение является препятствием :

- При необходимости совмещать разные версии компиляторов
- При необходимости подключать библиотеку на языке С

## Хорошая практика:

- Использовать extern "C"

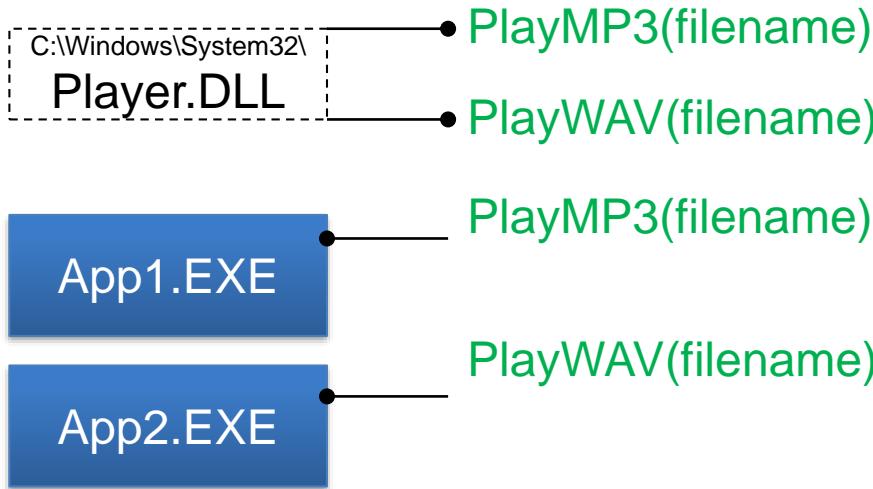
# Динамические библиотеки: Недостатки DLL

- В системе было установлено приложение App1.EXE и Player.DLL



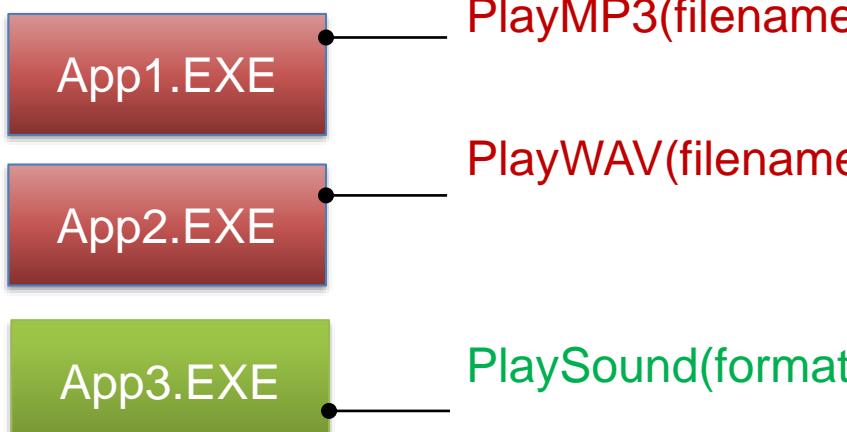
# Динамические библиотеки: Недостатки DLL

- Установлено приложение App1.EXE и Player.DLL первой версии
- Установлено приложение App2.EXE и Player.DLL следующей версии



# Динамические библиотеки: Недостатки DLL

- Установлено приложение App1.EXE и Player.DLL первой версии
- Установлено приложение App2.EXE и Player.DLL следующей версии
- Установлено приложение App3.EXE и Player.DLL новой версии

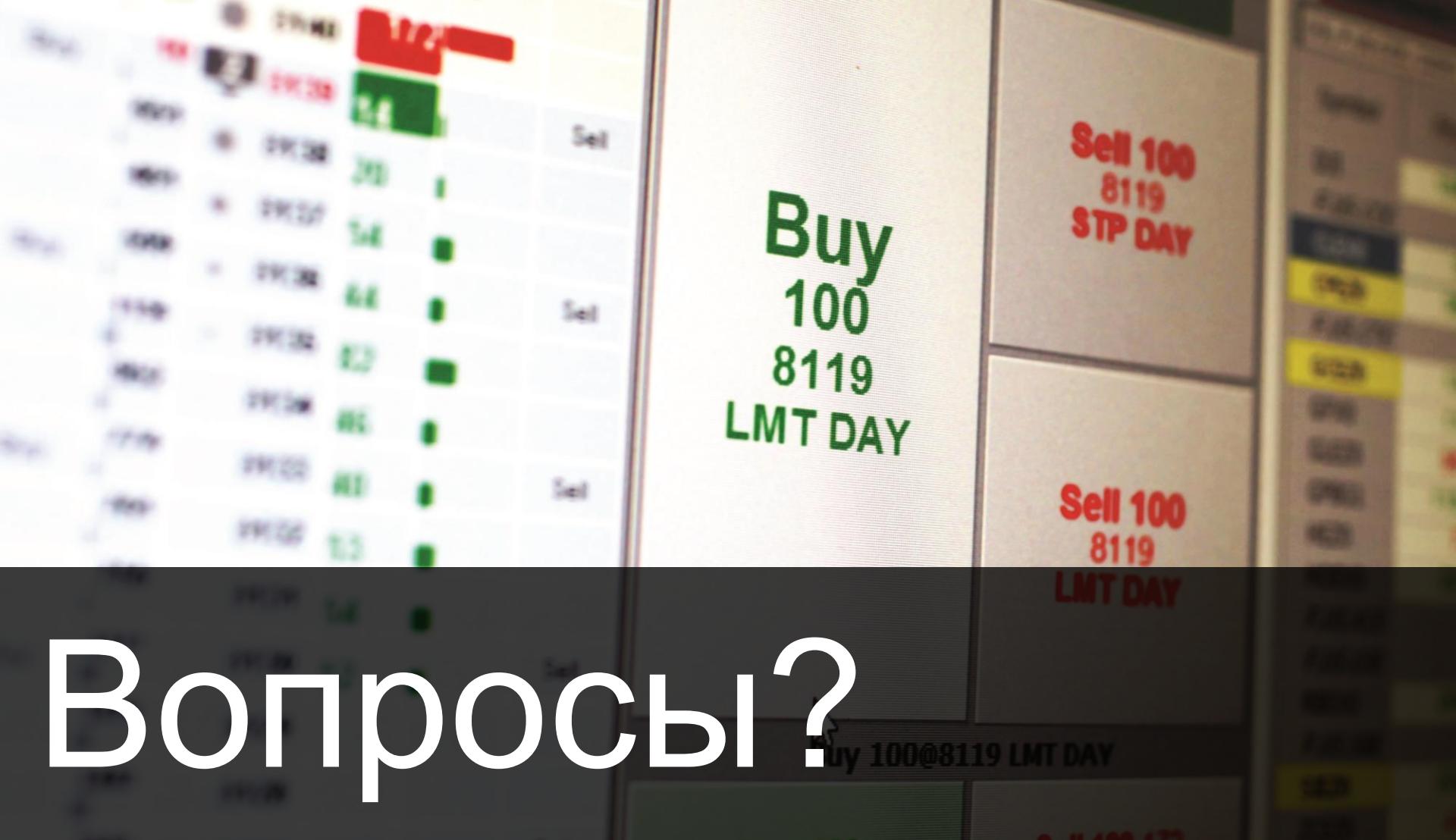


DLL HELL

# Динамические библиотеки: Как избежать DLL Hell

- Не устанавливать DLL для частного использования в общие папки
- Тщательное проектирование версионности на этапе разработки DLL
- Использование статически-линкуемых библиотек

# Вопросы?



# Динамические библиотеки: Использование dumpbin

```
D:\my_work\internship\2018\7. WinAPI и DLL\new\demo>z:  
Z:\>dumpbin /rawdata /section:.text dll_main.dll  
Microsoft (R) COFF/PE Dumper Version 14.16.27024.1  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Dump of file dll_main.dll  
File Type: DLL  
  
SECTION HEADER #1  
.text name  
A virtual size  
1000 virtual address (10001000 to 10001000)  
200 size of raw data  
400 file pointer to raw data  
0 file pointer to relocation table  
0 file pointer to line numbers  
0 number of relocations  
0 number of line numbers  
60000020 flags  
Code  
Execute Read  
  
RAW DATA #1  
10001000: 55 8B EC B8 00 20  
0 5D C3  
  
Summary  
1000 .text  
  
Z:\>dumpbin /disasm /section:.text dll_main.dll  
Microsoft (R) COFF/PE Dumper Version 14.16.27024.1  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Dump of file dll_main.dll  
File Type: DLL  
  
SECTION HEADER #1  
.text name  
A virtual size  
1000 virtual address (10001000 to 10001000)  
200 size of raw data  
400 file pointer to raw data  
0 file pointer to relocation table  
0 file pointer to line numbers  
0 number of relocations  
0 number of line numbers  
60000020 flags  
Code  
Execute Read  
  
10001000: 55 push ebp  
10001001: 8B EC mov ebp,esp  
10001003: B8 00 20 00 10 mov eax,10002000h  
10001008: 5D pop ebp  
10001009: C3 ret  
  
Summary  
1000 .text
```

# Динамические библиотеки: Использование dumpbin

calc.cpp (D:\Projects\ExplicitLinkage) - GVIM1

```
1 #include <iostream>
2 #include "Windows.h"
3
4 typedef int (*CalcFuncPtr)(int a, int b);
5
6 int main() {
7     HMODULE h = ::LoadLibraryA("math.dll");
8
9     if (h == NULL)
10    {
11        std::cout << "Cannot load library math.dll" << std::endl;
12        return -1;
13    }
14
15    CalcFuncPtr calcSum =
16        (CalcFuncPtr) ::GetProcAddress(h, "CalculateSum");
17    CalcFuncPtr calcDiff =
18        (CalcFuncPtr) ::GetProcAddress(h, "CalculateDif");
19
20    std::cout << calcSum(10, 15) << std::endl;
21    std::cout << calcDiff(25, 12) << std::endl;
22
23    ::FreeLibrary(h);
24    return 0;
25 }
```

D:\Projects\ExplicitLinkage>cl /c calc.cpp

Microsoft (R) C/C++ Optimizing Compiler Version 19.16.27034 for x86  
Copyright (C) Microsoft Corporation. All rights reserved.

calc.cpp

C:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\VC\

D:\Projects\ExplicitLinkage>link calc.obj

Microsoft (R) Incremental Linker Version 14.16.27034.0  
Copyright (C) Microsoft Corporation. All rights reserved.

D:\Projects\ExplicitLinkage>calc.exe

Cannot load library math.dll

# Динамические библиотеки: Использование dumpbin

math.cpp (D:\Projects\ExplicitLinkage) - GVIM1

```
1 extern "C"
2 __declspec(dllexport) int CalculateSum(int a, int b)
3 {
4     return a + b;
5 }
6
7 extern "C"
8 __declspec(dllexport) int CalculateDiff(int a, int b)
9 {
10    return a - b;
11 }
```

C:\WINDOWS\system32\cmd.exe [running] - GVIM1

Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Projects\ExplicitLinkage>cl /c math.cpp

Microsoft (R) C/C++ Optimizing Compiler Version 19.16.27034 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

math.cpp

D:\Projects\ExplicitLinkage>link /DLL /NOENTRY math.obj

Microsoft (R) Incremental Linker Version 14.16.27034.0

Copyright (C) Microsoft Corporation. All rights reserved.

Creating library math.lib and object math.exp

D:\Projects\ExplicitLinkage>

D:\Projects\ExplicitLinkage>calc.exe

25

13

# Динамические библиотеки: Использование dumpbin

```
D:\Projects\ExplicitLinkage>dumpbin /rawdata /section:.text math.dll
Microsoft (R) COFF/PE Dumper Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

Dump of file math.dll

File Type: DLL

SECTION HEADER #1

```
.text name
1B virtual size
1000 virtual address (10001000 to 1000101A)
200 size of raw data
200 file pointer to raw data (00000200 to 000003FF)
 0 file pointer to relocation table
 0 file pointer to line numbers
 0 number of relocations
 0 number of line numbers
```

60000020 flags

```
  Code
  Execute Read
```

RAW DATA #1

```
10001000: 55 8B EC 8B 45 08 03 45 0C 5D C3 CC CC CC CC U.i.E..E.]Ãiiiiii
10001010: 55 8B EC 8B 45 08 2B 45 0C 5D C3           U.i.E.+E.]Ã
```

Summary

```
1000 .text
```

# Динамические библиотеки: Использование dumpbin

```
D:\Projects\ExplicitLinkage>dumpbin /disasm /section:.text math.dll
Microsoft (R) COFF/PE Dumper Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

Dump of file math.dll

File Type: DLL

SECTION HEADER #1

```
.text name
 1B virtual size
 1000 virtual address (10001000 to 1000101A)
 200 size of raw data
 200 file pointer to raw data (00000200 to 000003FF)
 0 file pointer to relocation table
 0 file pointer to line numbers
 0 number of relocations
 0 number of line numbers
60000020 flags
  Code
 Execute Read
```

10001000: 55	push	ebp
10001001: 8B EC	mov	ebp,esp
10001003: 8B 45 08	mov	eax,dword ptr [ebp+8]
10001006: 03 45 0C	add	eax,dword ptr [ebp+0Ch]
10001009: 5D	pop	ebp
1000100A: C3	ret	
1000100B: CC	int	3
1000100C: CC	int	3
1000100D: CC	int	3
1000100E: CC	int	3
1000100F: CC	int	3
10001010: 55	push	ebp
10001011: 8B EC	mov	ebp,esp
10001013: 8B 45 08	mov	eax,dword ptr [ebp+8]
10001016: 2B 45 0C	sub	eax,dword ptr [ebp+0Ch]
10001019: 5D	pop	ebp
1000101A: C3	ret	

Summary

1000 .text

# Динамические библиотеки: Использование dumpbin

```
D:\Projects\ExplicitLinkage>dumpbin /exports math.dll
Microsoft (R) COFF/PE Dumper Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
D:\Projects\ExplicitLinkage>dumpbin /imports calc.exe | grep Calculate
```

Dump of file math.dll

File Type: DLL

Section contains the following exports for math.dll

```
00000000 characteristics
FFFFFFF time date stamp
0.00 version
    1 ordinal base
    2 number of functions
    2 number of names
```

ordinal	hint	RVA	name
---------	------	-----	------

1	0	00001010	CalculateDiff
2	1	00001000	CalculateSum

Summary

1000	.rdata
1000	.text

# Динамические библиотеки: Использование dumpbin

calc.cpp (D:\Projects\ImplicitLinkage) - GVIM1

```
1 #include <iostream>
2 #include "math.h"
3
4 int main()
5 {
6     std::cout << CalculateSum(10, 15) << std::endl;
7     std::cout << CalculateDif(25, 12) << std::endl;
8     return 0;
9 }
```

```
D:\Projects\ImplicitLinkage>cl /c calc.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.16.27034 for x86
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
calc.cpp
C:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\VC\Tools\MSVC\14.16.27023\include\xlocal
```

```
D:\Projects\ImplicitLinkage>link calc.obj
Microsoft (R) Incremental Linker Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
calc.obj : error LNK2019: unresolved external symbol __imp__CalculateSum referenced in function _main
calc.obj : error LNK2019: unresolved external symbol __imp__CalculateDif referenced in function _main
calc.exe : fatal error LNK1120: 2 unresolved externals
```

math.h (D:\Projects\ImplicitLinkage) - GVIM1

```
1 #ifndef MATHAPI
2 #define MATHAPI extern "C" __declspec(dllexport)
3 #endif
4
5 MATHAPI int CalculateSum(int a, int b);
6 MATHAPI int CalculateDif(int a, int b);
```

# Динамические библиотеки: Использование dumpbin

```
math.cpp (D:\Projects\ImplicitLinkage) - GVIM1
1 #define MATHAPI extern "C" __declspec(dllexport)
2 #include "math.h"
3
4 int CalculateSum(int a, int b)
5 {
6     return a + b;
7 }
8
9 int CalculateDif(int a, int b)
10 {
11     return a - b;
12 }
```

```
D:\Projects\ImplicitLinkage>cl /c math.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.16.27034 for x86
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
math.cpp
```

```
D:\Projects\ImplicitLinkage>link /DLL /NOENTRY math.obj
Microsoft (R) Incremental Linker Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
Creating library math.lib and object math.exp
```

```
math.h (D:\Projects\ImplicitLinkage) - GVIM1
1 #ifndef MATHAPI
2 #define MATHAPI extern "C" __declspec(dllimport)
3 #endif
4
5 MATHAPI int CalculateSum(int a, int b);
6 MATHAPI int CalculateDif(int a, int b);
```

```
D:\Projects\ImplicitLinkage>link calc.obj math.lib
Microsoft (R) Incremental Linker Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
D:\Projects\ImplicitLinkage>calc.exe
25
13
```

# Динамические библиотеки: Использование dumpbin

```
D:\Projects\ImplicitLinkage>dumpbin /exports math.dll
Microsoft (R) COFF/PE Dumper Version 14.16.27034.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
D:\Projects\ImplicitLinkage>dumpbin /imports calc.exe | grep Calculate
0 CalculateDif
1 CalculateSum
```

Dump of file math.dll

File Type: DLL

Section contains the following exports for math.dll

```
00000000 characteristics
FFFFFFF time date stamp
0.00 version
1 ordinal base
2 number of functions
2 number of names
```

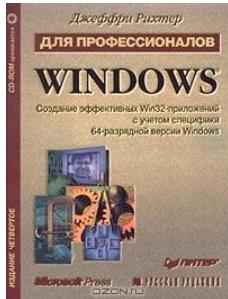
ordinal	hint	RVA	name
---------	------	-----	------

1	0	00001010	CalculateDif
2	1	00001000	CalculateSum

Summary

1000	.rdata
1000	.text

# Дополнительные ресурсы



**Джеффри Рихтер.** Windows для профессионалов.  
Создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows



<https://docs.microsoft.com/en-us/windows/desktop/Dlls/about-dynamic-link-libraries>



CppCon 2017: **James McNeillis**  
“Everything You Ever Wanted to Know about DLLs”

# Вопросы?

