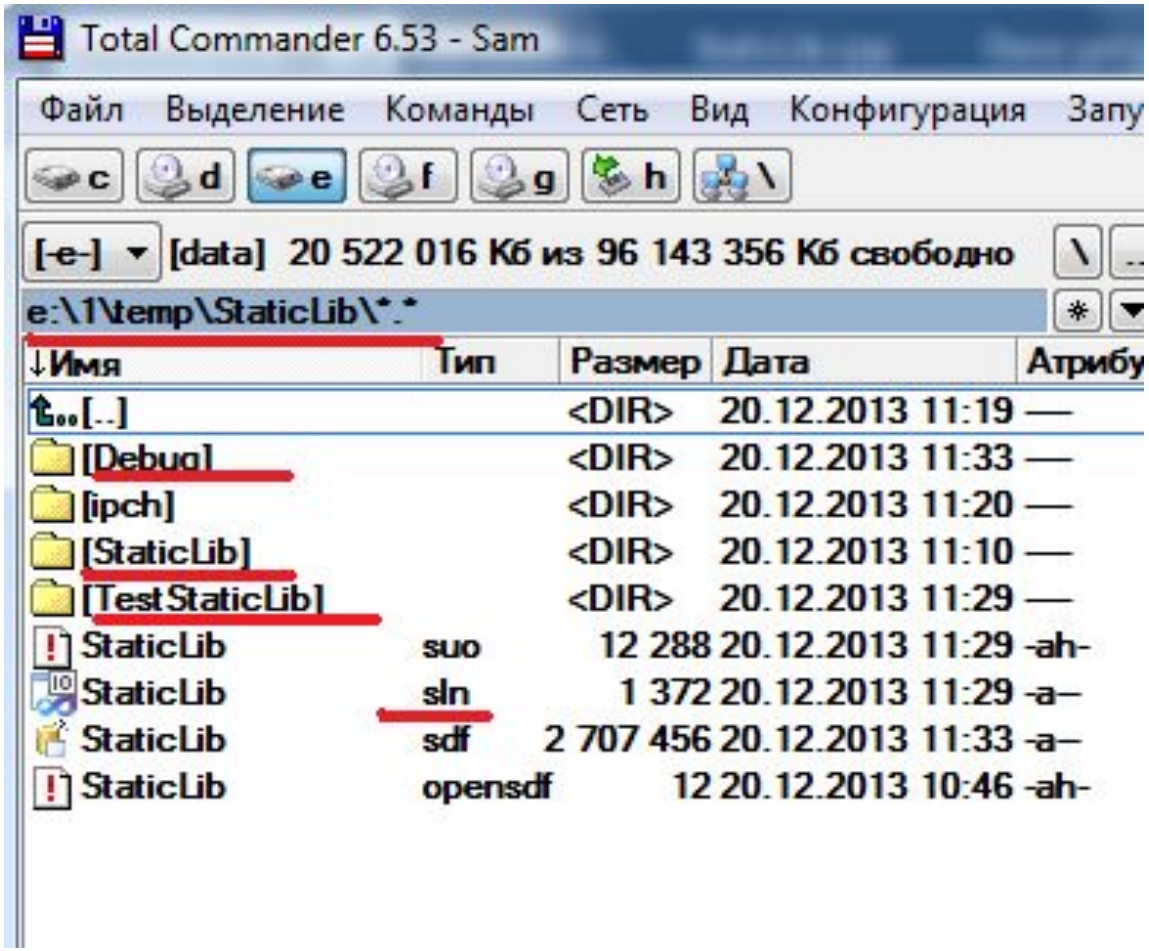
Каталоги с Решением (solution), содержащим оба проекта.

Имя Решения обычно совпадает с именем первого созданного в нем Проекта (Project).

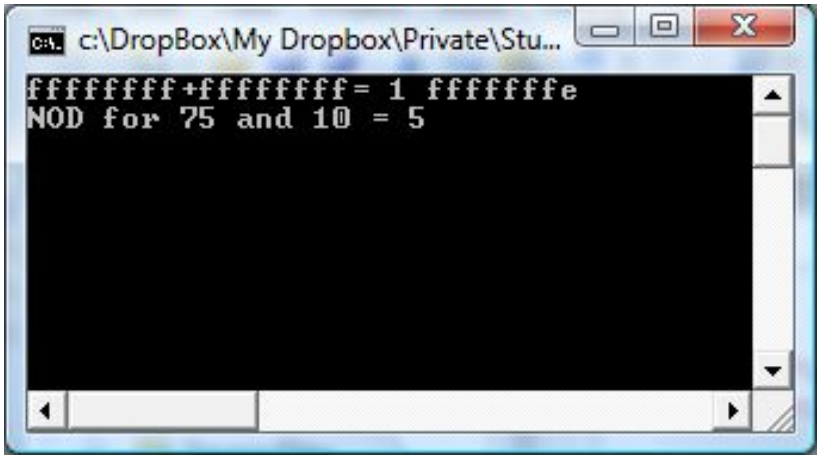
Имя Решения:
StaticLib

Имя проекта со статической библиотекой:
StaticLib

Имя проекта с главной Программой, которая тестирует функции из статической библиотеки:
TestStaticLib



Главная программа для использования статической библиотеки . Результаты работы программы: на экран или в файл.



Запуск программы из командной строки:

TestStaticLib.exe > output.txt

Файл с результатами output.txt:

ffffffff+ffffffff= 1 fffffffe
NOD for 75 and 10 = 5

Достоинства и недостатки статических библиотек

Достоинства:

- просто использовать;
- исполняемый файл один (.exe).

Недостатки:

- зависит от среды разработки;
- загружается в память с каждым экземпляром запущенного приложения;
- при изменении кода библиотеки необходима компоновка всех приложений, которые используют библиотеку.

Итоги:

- библиотеки применяются для повторного использования кода;
- статическая библиотека - это библиотека объектных модулей;
- для использования статической библиотеки необходимо иметь саму библиотеку (.lib) в формате среды (IDE), в которой она будет использоваться и заголовочный файл (.h) с определением заголовков функций библиотеки;
- отсутствуют накладные затраты, связанные с использованием динамических библиотек.

Лекции 5, 6

Динамические библиотеки. Командные файлы. Работа с внешними устройствами.

План лекций № 5, 6

- Создание динамических библиотек;
- Использование динамических библиотек. Статический режим;
- Использование динамических библиотек. Динамический режим;
- Пример создания и использования;
- Преимущества и недостатки Dll;
- DllMain и проверка целостности;
- Командные файлы
- Работа с внешними устройствами.

Динамические библиотеки

Динамические библиотеки (Dynamic Link Library - DLL)

Загружаются одновременно с программой (статическая загрузка) или во время ее выполнения по мере надобности (динамическая загрузка).

==========

Функция, которая экспортируется (внешняя функция) - это функция, которая входит в состав DLL, и которую могут использовать внешние программы.

(в статических библиотеках - все функции экспортируются).

Для обозначения внешних функций используется директива:

__declspec (dllexport)

Динамические библиотеки

Функция, которая импортируется - это функция из DLL, которая вызывается (используется) в другой программе.

Функции, которые импортируются, обозначаются директивой:

__declspec (dllimport)

(пример: страны экспортеры и импортеры)

Таким образом, одна и та же функция:

- внутри самой DLL является функцией, которая экспортируется;
- для главной программы - функцией, которая импортируется.

Внутренняя функция библиотеки может быть вызвана только функциями внутри библиотеки.

Обозначение функций

Исходя из вышесказанного, в файле заголовков (.h) информация о внешних функциях должна быть разной:

- для самой библиотеки .dll - экспорт,
- а для главной программы - импорт!

```
// объявление функции внутри библиотеки DLL
__declspec (dllexport)  заголовок  функции1
__declspec (dllexport)  заголовок  функции2
…
==========================================================
```

```
// объявление функции в главной программе
__declspec (dllimport)  заголовок  функции1
__declspec (dllimport)  заголовок  функии2
…
```

- Решается путем использования универсального заголовочного файла =>

Универсальный заголовочный файл **(universal.h)**

```
#ifndef   _UNIVERSAL_H
  #define  _UNIVERSAL_H
#ifdef    _STATIC
          #define PREFIX
#else
    #ifdef  _USRDLL
          #define PREFIX   __declspec(dllexport)
    #else
          #define PREFIX   __declspec(dllimport)
    #endif
#endif
PREFIX unsigned int  __stdcall AddWithCarry( unsigned int , unsigned int, unsigned int*);
PREFIX void  __stdcall NOD( unsigned int a, unsigned int b, unsigned int* r);
#endif
```

Динамические библиотеки

1 Статическая загрузка (загрузка во время загрузки приложения, которое использует DLL) - если нет необходимой DLL - приложение не начнет выполняться;

2 Динамическая загрузка (загрузка и выгрузка по мере необходимости во время выполнения приложения, которое использует DLL) - загружаются только те DLL, функции из которых будут вызываться; после окончания использования память можно освободить, не дожидаясь окончания работы главной программы.

Динамические библиотеки

Все модули операционной системы делятся на 2 класса: ядра и пользователя

- *Ядро – модули, необходимые для работы любой программы, постоянно в памяти, загружаются на этапе загрузки ОС, имеют 0 приоритет, т.е. работают в режиме ядра;*
- *Утилиты (службы) – специализированные модули, могут загружаться и выгружаться, работают в режиме пользователя*

Объект ядра - блок памяти в адресном пространстве ядра, доступный только самому ядру. Программам пользователя этот участок памяти напрямую не доступен.

В момент загрузки DLL проецируется в адресное пространство вызвавшего ее процесса, а в адресном пространстве ядра создается соответствующий объект ядра (структура данных).

Существует набор функций (из системной библиотеки WIN32 API), которые могут работать с объектами ядра. Обращаясь к этим функциям, программы могут получить данные из объектов ядра.

Динамические библиотеки

Создание:

1. Выбрать проект типа Visual C++-> Win32; в ApplicationSettings выбрать DLL
2. Добавить в проект **универсальный** заголовочный файл;
3. Добавить в проект файл с текстом функций;
4. Обратить внимание на наличие файла dllmain.cpp;
5. Построить проект. В результате будут получены 2 файла: <имя>.lib и <имя>.dll (.exe файл создан не будет!!);

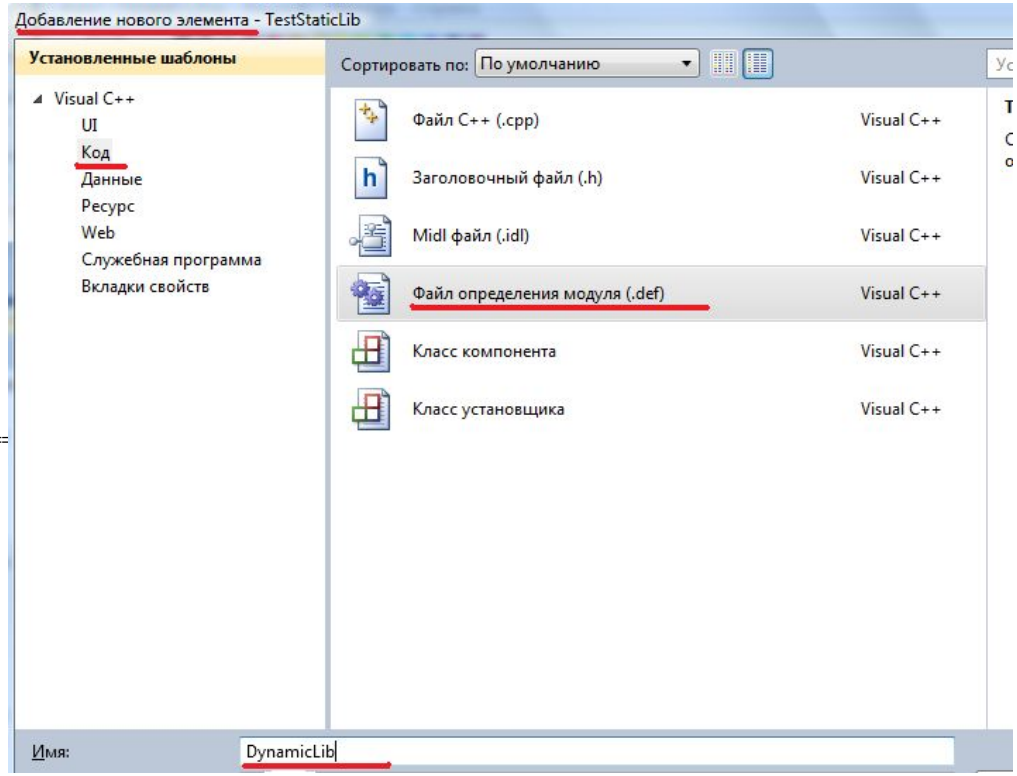
Использование:

1. *Использования динамической библиотеки в режимах статической и динамической загрузки.*

DEF файл

Файл с расширением .def добавляется в проект DLL для сохранения возможности обращения к функциям по их именам без преобразования при динамической загрузке DLL.

LIBRARY “Имя библиотеки"
EXPORTS
Имя функции 1
Имя функции 2
…



DynamicLib.def:

LIBRARY "DynamicLib"
EXPORTS
AddWithCarry
NOD

Библиотека системных функций WIN32 API - интерфейс между ОС и приложениями пользователя

WIN32 API
(Windows Application Programming Interface) :

- набор функций, реализующих выполнение основных функций ОС;
- 3 основных модуля – kernel32.dll, gdi32.dll, user32.dll

// где читать - MSDN, Рихтер.

Правила использования функций WIN32 API

1. Необходимо подключить заголовочный файл Windows.h
2. Все функции имеют соглашения по вызову __stdcall (WINAPI)
3. Все типы и константы Windows задаются заглавными буквами, например, DWORD – unsigned int , WORD – short, HMODULE - int, INVALID_HANDLE_VALUE.
4. Каждое слово в имени функции начинается с заглавной буквы, например, LoadLibrary.
5. Функция может завершиться успешно или неуспешно – BOOL (0 – false, 1 – true…).
6. Если она возвращает дескриптор, то в случае ошибки:

   0 или INVALID_HANDLE_VALUE.

Функции для работы с DLL в режиме динамической загрузки

- HMODULE WINAPI LoadLibrary(LPCTSTR *lpFileName*);
- BOOL WINAPI FreeLibrary(HMODULE *hModule*);
- FARPROC WINAPI GetProcAddress(HMODULE *hModule*, LPCSTR *lpProcName*);

Алгоритм поиска DLL

- Каталог, в котором находится исполняемый модуль текущего процесса.
- Текущий каталог (GetCurrentDirectory).
- Системный каталог Windows. Путь к этому каталогу извлекается с помощью функции GetSystemDirectory.
- Каталог Windows. Путь к этому каталогу извлекается с помощью функции GetWindowsDirectory.
- Каталоги, указанные в переменной окружения PATH.

Главная программа для DLL в режиме динамической загрузки

```
#include "stdafx.h"
#include <windows.h>
#include <stdio.h>
#include “universal.h“

typedef   unsigned int  (__stdcall *ADDWITHCARRY)( unsigned int , unsigned int, unsigned int*);
typedef  void           (__stdcall *NODD)          ( unsigned int , unsigned int, unsigned int*);

int _tmain(int argc, _TCHAR* argv[])
{
    unsigned int first=0xffffffff,second=0xffffffff, result, carry;
    HMODULE h=LoadLibrary(_T("Lecture_dynamic_library.dll"));
    if (h==NULL)
      _tprintf(_T("Library not found"));
    else
    {ADDWITHCARRY adr1=(ADDWITHCARRY)GetProcAddress(h,"AddWithCarry");
     NODD adr2=(NODD)GetProcAddress(h,"NOD");

     carry=adr1(first,second,&result);
     _tprintf(_T("%x+%x= %x %x\n"),first,second,carry,result);

     adr2(75,10,&result);
     _tprintf(_T("NOD for %u and %u = %u"),75,10,result);
     FreeLibrary(h);
    }
    return 0;
}
```

Рекомендации по отладке ДЛЛ

- В одном Решении и проект для создания ДЛЛ и проект для отладки функций из библиотеки;
- Так как заголовочный файл должен быть общим для всех проектов Properties → C/C++ →Additional Include Directories ..\Имя проекта ДЛЛ

Файл Dllmain.cpp. Точка входа в ДЛЛ.

```
BOOL APIENTRY DllMain( HMODULE hModule, DWORD  ul_reason_for_call, LPVOID lpReserved )
{
    switch (ul_reason_for_call)
    {
      case DLL_PROCESS_ATTACH:
            if ( ПРОВЕРКА_ЦЕЛОСТНОСТИ==false)
                return false;
            else
                return true;
      case DLL_THREAD_ATTACH:
      case DLL_THREAD_DETACH:
      case DLL_PROCESS_DETACH:
      break;
    }
   return TRUE;
}
```

Достоинства и недостатки DLL

Достоинства:

- в память загружаются только один раз независимо от числа приложений, которые их используют;
- могут использоваться в среде, отличной от среды разработки (C++ →C#, Delphi);
- в случае изменения ДЛЛ не требуется перекомпоновка приложений (так поставляются Service Pack'и);
- можно выгружать, освобождая адресное пространство.

Недостатки:

- требуется дополнительный файл (в дополнение к главному .exe);
- легко изменить функциональность(подменить), если не предусмотрен контроль целостности;
- требуют специальных знаний для использования.

Введение в командные файлы (.cmd, .bat)

Используются, если необходимо стандартно выполнять более одного действия, например, последовательно запускать несколько программ; запускать или не запускать программы в зависимости от результатов работы другой программы, …

Состоят из:

- Системных команд;
- Команд запуска приложений;
- Директив для условного запуска и организации циклов.

Информация о системных командах:

Кнопка «Пуск»->Выполнить: cmd.exe → help>help.txt → exit
В Windows Commander: help>help.txt

Введение в командные файлы (.cmd, .bat)

Примеры системных команд:

copy /v /y источник результат

где

источник - имя копируемого файла;
результат - имя для конечного файла.

/v Проверка правильности копирования файлов.
/y Подавление запроса подтверждения на перезапись существующего
/-y Обязательный запрос подтверждения на перезапись существующего конечного файла.

Пример использования:

copy /v /y help.txt 1.txt

вызов помощи по конкретной команде, например по copy:

help copy

Введение в командные файлы (.cmd, .bat)

Пример .bat файла:

test.bat

==========

```
copy  /v /y %1 %2
rem del %1
rem mspaint.exe
```

==========

Запуск командного файла из командной строки (при условии, что исходный файл лежит в текущем каталоге):

```
test.bat help.txt 1.txt
```

Среда Visual Studio и командные файлы

Когда надо задавать?

- если для выполнения проекта необходимые файлы, которые надо откуда-то скопировать;
- если перед компоновкой (построением) необходимо откуда-то взять библиотеки;
- *если после создания исполняемого файла его необходимо куда-то перенести, или выполнить над ним какие-то операции, например, сформировать контрольную сумму*
- Задавать можно только .bat файл.

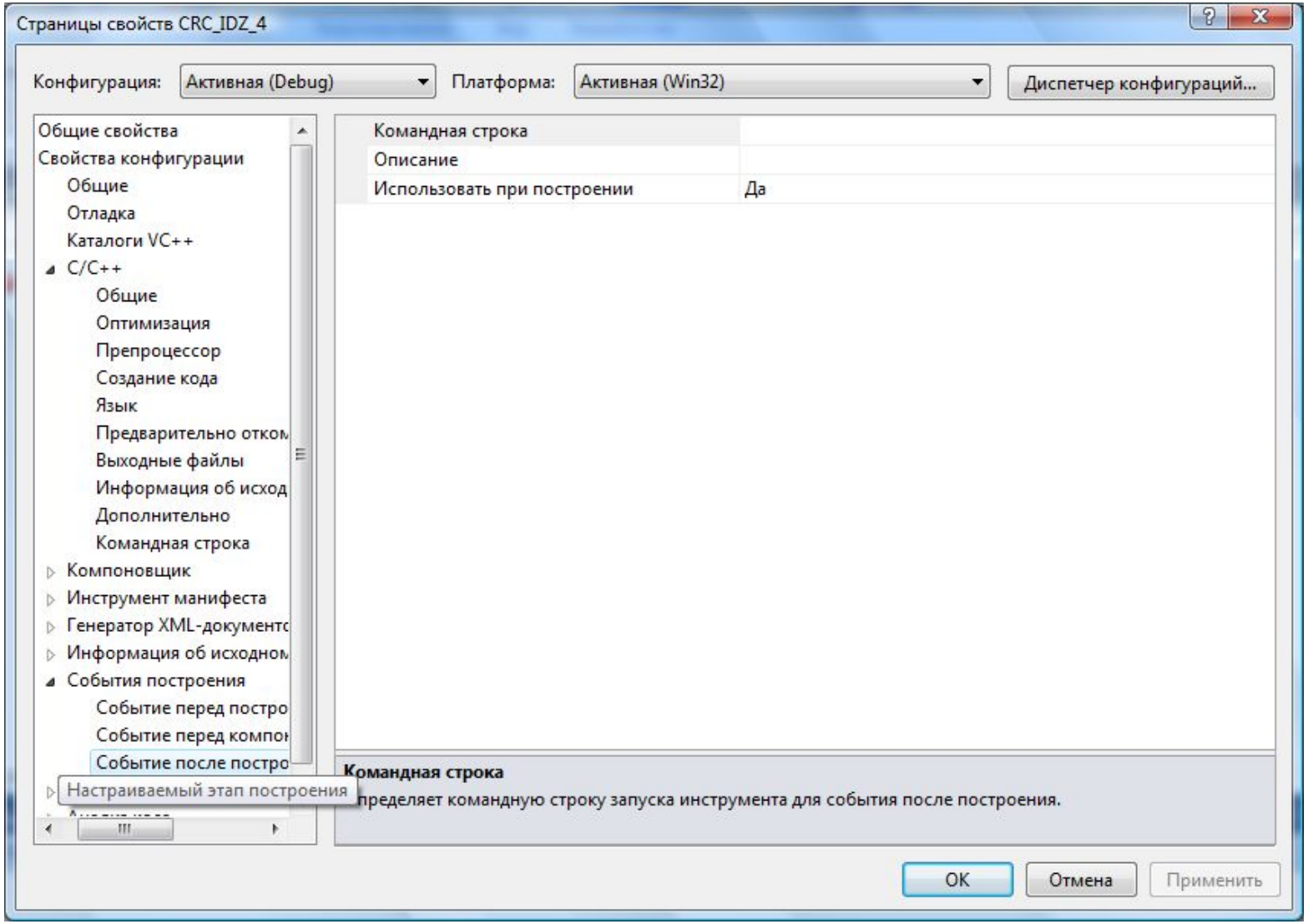
Среда Visual Studio и командные файлы

Где надо задавать?

- Properties->Build Events
- Pre-Build Event перед построением
- Pre-Link Event перед компоновкой
- Post-Build Event после построения

  - CommandLine (командная строка)
  - Description (описание действий)
  - Excluded From Build (использовать ли при текущем построении?)

Среда Visual Studio и командные файлы



Среда Visual Studio и командные файлы

Как задать имя командного файла?

1. Выбрать CommandLine и задать командный файл, который надо выполнить.
2. При задании использовать макросы, которые определены при создании solution в VS

Среда Visual Studio и командные файлы. Макросы

Имя	Значение
$(ConfigurationName)	*Debug* или *Release*
$(OutDir)	Каталог для результата (с или без \ зависит от версии VS)
$(ProjectDir)	Каталог с проектом (. vcproj) (с или без \ зависит от версии VS)
$(SolutionDir)	Каталог с решением (. sln) (с или без \ зависит от версии VS)

Среда Visual Studio и командные файлы. Итоги

- Командные файлы помогают выполнять фиксированную последовательность действий;
- есть помощь по использованию системных функций;
- командные файлы поддерживаются всеми версиями ОС и не нуждаются в установке дополнительного ПО.

Управление внешними устройствами

3 уровня:

1. функции языков программирования(scanf, printf, fscanf, fprintf, …)
2. **функции операционной системы (системные вызовы)**

- функции для управления стандартными устройствами (консоль : клавиатура + монитор)

- функции для управления файлами

3. функции физических драйверов

Управление внешними устройствами (стандартные устройства)

Создание и удаление консоли
BOOL WINAPI AllocConsole(void
BOOL WINAPI FreeConsole(void)

Имена стандартных устройств:
STD_INPUT_HANDLE
STD_OUTPUT_HANDLE
STD_ERROR_HANDLE

Определение дескриптора консоли:
HANDLE GetStdHandle (Имя устройства);

// если ошибка - INVALID_HANDLE_VALUE

Операции ввода – вывода
BOOL ReadConsole (дескр, Buf, size, &dwCount, 0);
BOOL WriteConsole (дескр, Buf, size, &dwCount, 0);

Управление внешними устройствами (стандартные устройства)

```
// Вывести строку-приглашение. Вводить и выводить строки до тех пор пока не будет
// введена пустая строка
HANDLE hIn = GetStdHandle (STD_INPUT_HANDLE);
HANDLE hOut = GetStdHandle (STD_OUTPUT_HANDLE);
if (hIn != INVALID_HANDLE_VALUE && hOut != INVALID_HANDLE_VALUE)
{
    DWORD dwCount;
    TCHAR Buf [80];
    TCHAR Prompt [] = _T("Input Text\n");
    BOOL b = WriteConsole (hOut, Prompt, sizeof (Prompt)/ sizeof (TCHAR), &dwCount, 0);
    while (1)
    {
     ReadConsole (hIn, Buf, sizeof (Buf), &dwCount, 0);
     WriteConsole (hOut, Buf, dwCount, &dwCount, 0);
     if (Buf [0] = = _T('\r') && Buf [1] = = _T('\n')
          break;
    }
}
```

Недостаток – есть эхо

РЕЖИМЫ РАБОТИ КОНСОЛИ

BOOL **GetConsoleMode**(HANDLE hConsoleHandle, LPDWORD lpMode);

ENABLE_ECHO_INPUT // отображение символов, которые вводятся;
ENABLE_LINE_INPUT // ждет введа всей строки;
ENABLE_MOUSE_INPUT// реагирует на перемещение мышки;
ENABLE_INSERT_MODE // включен режим вставки (если
// исключен, то используется режим замены)

BOOL WINAPI **SetConsoleMode**(HANDLE hConsoleHandle, DWORD lpMode);

Режимы работы консоли. Пример

Пример1. Отключить эхо и ожидания ввода всей строки

```
HANDLE hConsoleHandle = GetStdHandle (STD_INPUT_HANDLE);
DWORD dwState, dwOldState;
GetConsoleMode(hConsoleHandle, &dwState);
dwOldState= dwState;
dwState= dwState & ~( ENABLE_ECHO_INPUT| ENABLE_LINE_INPUT);
SetConsoleMode(hConsoleHandle, dwState);
….

SetConsoleMode(hConsoleHandle, dwOldState);
```

Файлы

```
HANDLE WINAPI CreateFile(
      LPCTSTR               lpFileName,
      DWORD                 dwDesiredAccess,
      DWORD                 dwShareMode,                    // 0
      LPSECURITY_ATTRIBUTES lpSecurityAttributes,           // 0
      DWORD                 dwCreationDisposition,
      DWORD                 dwFlagsAndAttributes,           // 0
   HANDLE          hTemplateFile                    // 0
);
```

==========

lpFileName: MAX_PATH (260) – WinDefs.h (C:\ + 256 символів)

dwDesiredAccess GENERIC_READ, GENERIC_WRITE

dwShareMode FILE_SHARE_READ, FILE_SHARE_WRITE

dwCreationDisposition

CREATE_ALWAYS Creates a new file, always. If a file exists, the function overwrites the f.

CREATE_NEW Creates a new file. The function fails if a specified file exists.

OPEN_ALWAYS Opens a file, always. If a file does not exist, the function creates a f.

OPEN_EXISTING Opens a file. The function fails if the file does not exist.

Проверка целостности

Test.bat:

%1Add_Crc.exe %2

Запуск .bat файла из командной строки:

Test.bat C:\Study\ D:\Temp\lab1.dll

аналогичен следующему запуску Add_Crc.exe из командной строки:

C:\Study\Add_Crc.exe D:\Temp\lab1.dll

dllmain.cpp

```
#include "stdafx.h"
#include "tchar.h“
// After .dll will be created command file test.bat calls add_crc.exe.
//  test.bat is placed in current Project Directory.
// add_crc.exe adds CRC in the end of .dll.
// add_crc.exe takes 1 argument:  combination of path and name of .dll

// The CheckCRC does Cyclical Redundancy Check and compares result with last 4 bytes in the
// end of file. Returns false if DLL was substituted, otherwise returns true.

bool CheckCRC(HMODULE hMod)
{
DWORD crc=0,CRCtemplate;
DWORD High,Low, data,real;
TCHAR LibName[MAX_PATH];

GetModuleFileName(hMod,LibName,MAX_PATH);

HANDLE  hFile=CreateFile(LibName,GENERIC_READ,FILE_SHARE_READ,0,OPEN_EXISTING,0,0);
if(hFile==INVALID_HANDLE_VALUE)
{
_tprintf(_T("DLL not found during CheckCRC!"));  return 0;
}
```

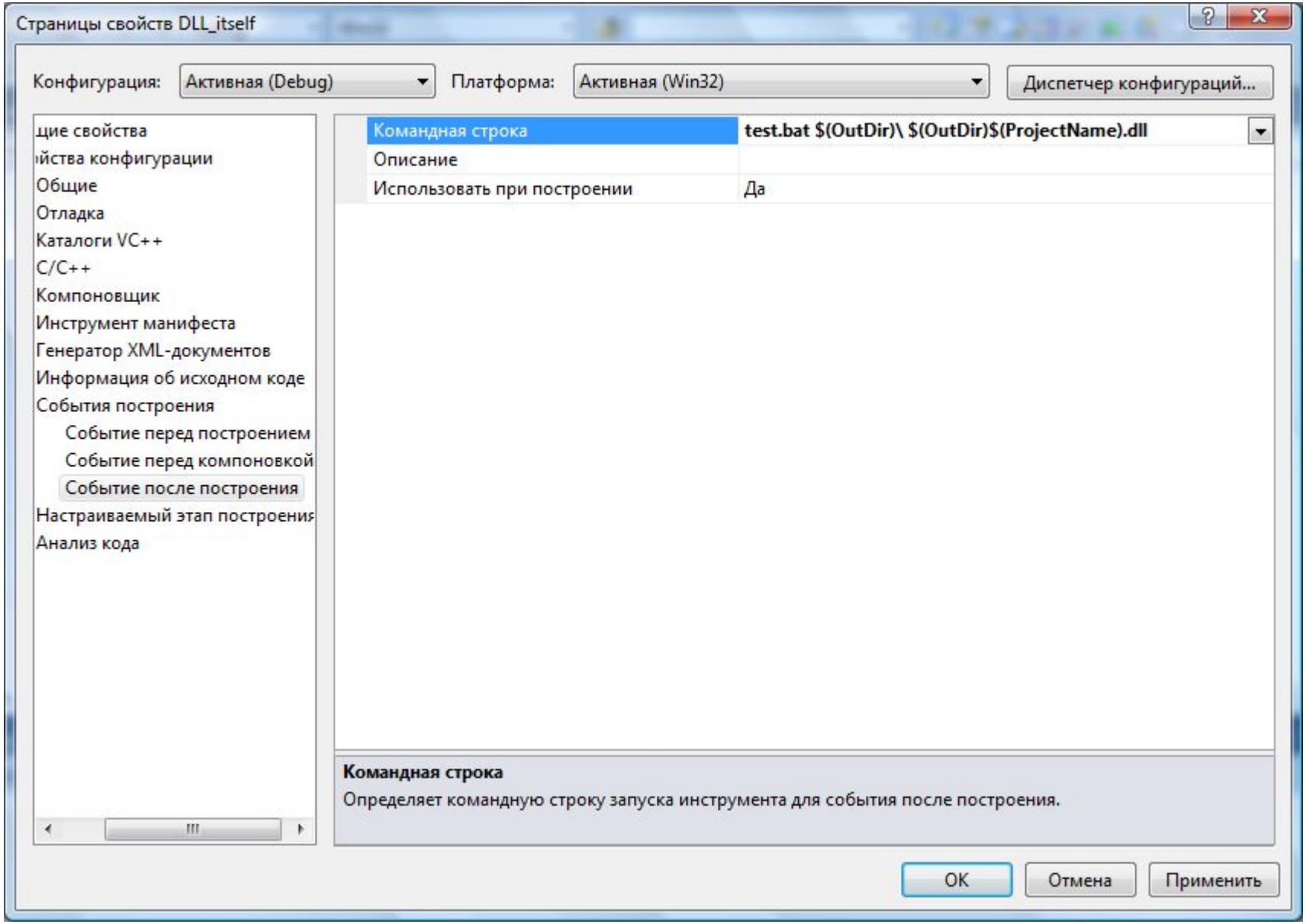
dllmain.cpp

```
// продолжение предыдущего слайда
Low=GetFileSize(hFile,&High);

int counter=(Low-4)/4;
int rem=Low%4;

for(int i=0;i<counter;i++)
{
ReadFile(hFile,&data,4,&real,0);
crc=(crc+data)%0xffff;
}
ReadFile(hFile,&data,rem,&real,0);
crc=(crc+data)%0xffff;

ReadFile(hFile,&CRCtemplate,4,&real,0);
CloseHandle(hFile);
if (CRCtemplate==crc)
return true;
else
return false;

}
```

Файл dllmain.cpp. Точка входа в ДЛЛ.

```
                                        //  продолжение предыдущего слайда
BOOL APIENTRY DllMain( HMODULE hModule,     DWORD  ul_reason_for_call,    LPVOID lpReserved  )
{
switch (ul_reason_for_call)
{
case DLL_PROCESS_ATTACH:
                        if ( !CheckCRC(hModule) )
                        {
                        _tprintf(_T("CheckCRC returns FALSE!\n"));
                        return false;
                        }
                        else
                        {
                        _tprintf(_T("CheckCRC returns TRUE!\n"));
                        return true;
                        }
case DLL_THREAD_ATTACH:
case DLL_THREAD_DETACH:
case DLL_PROCESS_DETACH:
break;
}
return TRUE;
}
```

СРЕДА VISUAL STUDIO И КОМАНДНЫЕ ФАЙЛЫ



Программа, загружающая DLL в динамическом режиме

```
……..
int _tmain(int argc, _TCHAR* argv[])
{

// If DLL was substituted LoadLibrary returns NULL

hLib=LoadLibrary(_T("DLL_itself.dll"));
if(hLib==NULL)
{
_tprintf(_T("No Library Loaded\n"));
_getch();
return -2;
}

……
```

Просмотр кода .dll