

Разработка и использование
библиотек.
Лекция 3 и 4

План лекции №3

- Назначение и классификация библиотек;
- Соглашения по вызову;
- Создание статических библиотек;
- Использование статических библиотек;
- Пример создания и использования;
- Преимущества и недостатки статических библиотек.

Назначение и классификация библиотек

Назначение:

Повторное использование кода.

Используются если:

- Функции из библиотеки будут использоваться в разных программах;
- разрабатывается большой проект, часть функций уже отлажена - эту часть помещают в библиотеку (время трансляции кода уменьшается).

Назначение и классификация библиотек

Классификация библиотек:

- библиотеки на языках программирования (библиотеки классов, шаблонов, функций...). Компилируются вместе с остальными исходными файлами проекта (*не будут рассматриваться*);
- библиотеки объектных модулей (статические библиотеки). Компилируются вместе с остальными объектными файлами проекта (*будут рассматриваться*);
- библиотеки исполняемых модулей (динамические библиотеки). Загружаются в память в момент запуска программы или во время ее исполнения, по мере необходимости (*будут рассматриваться*).

Соглашения по вызовам

- В одном приложении можно использовать библиотеки, написанные на разных языках;
- язык написания вызывающей программы может не совпадать с языком функций в библиотеке.

Соглашение определяет:

- порядок передачи параметров (с начала или конца списка);
- правила очистки стека (вызывающая программа, функция);
- зависимость имени функции от списка параметров.

Тип соглашения	Как задается	Параметры	Направление передачи пар.	Очистка стека	Имя функции
C++		Стек	<input type="checkbox"/>	программа	изменяется
stdcall	__stdcall	Стек	<input type="checkbox"/>	функция	изменяется
aastcall	__fastcall	Регистры (eax, edx, ecx) + стек	<input type="checkbox"/>	функция	изменяется
...					

Соглашения по вызовам

Заголовок функции:

Тип результата [Соглашение] Имя ([Параметры]) ;

Например:

```
int Sum(int , int);
```

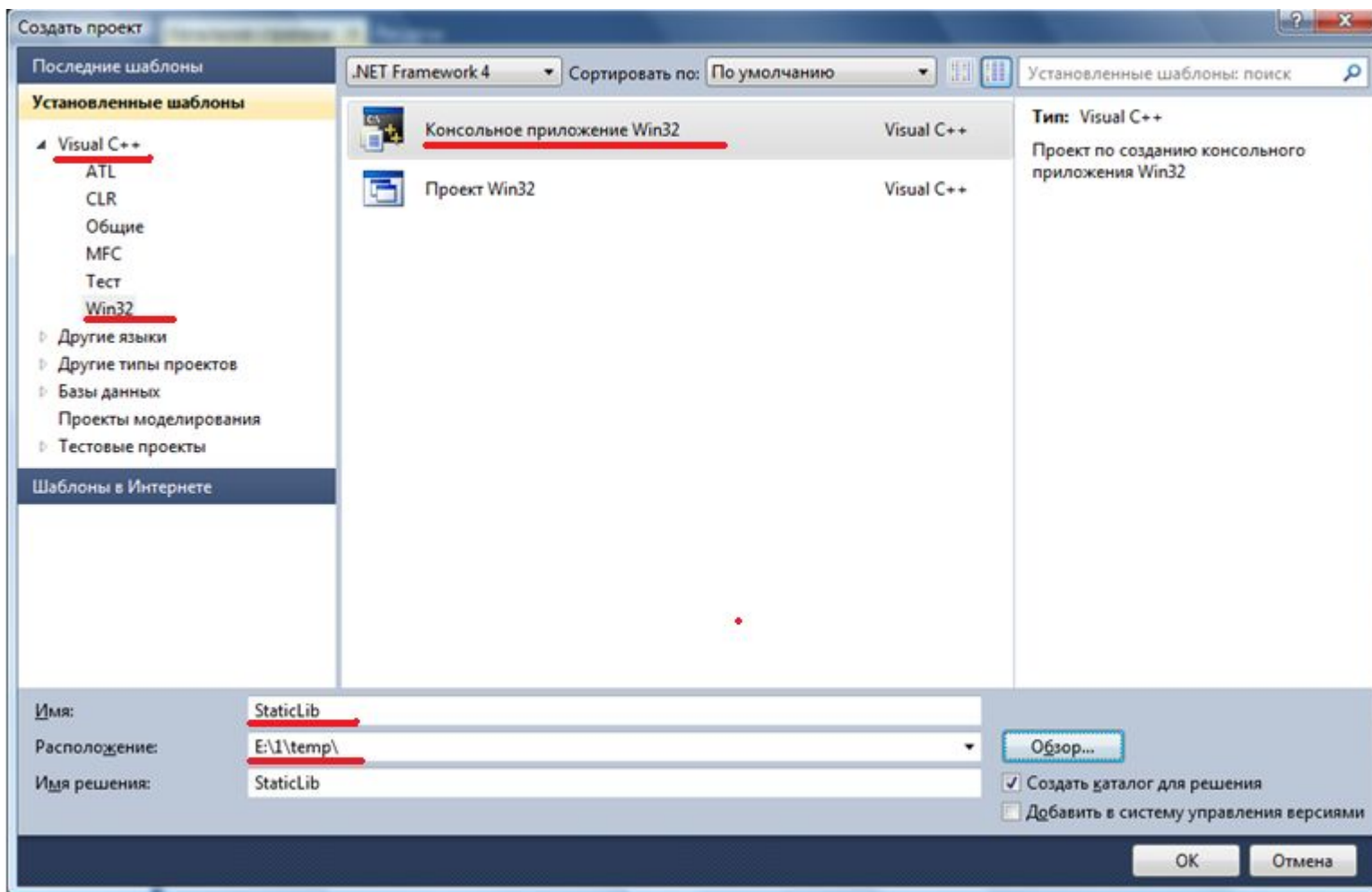
```
void __stdcall AddValues(int , int , int*);
```

Создание статических библиотек

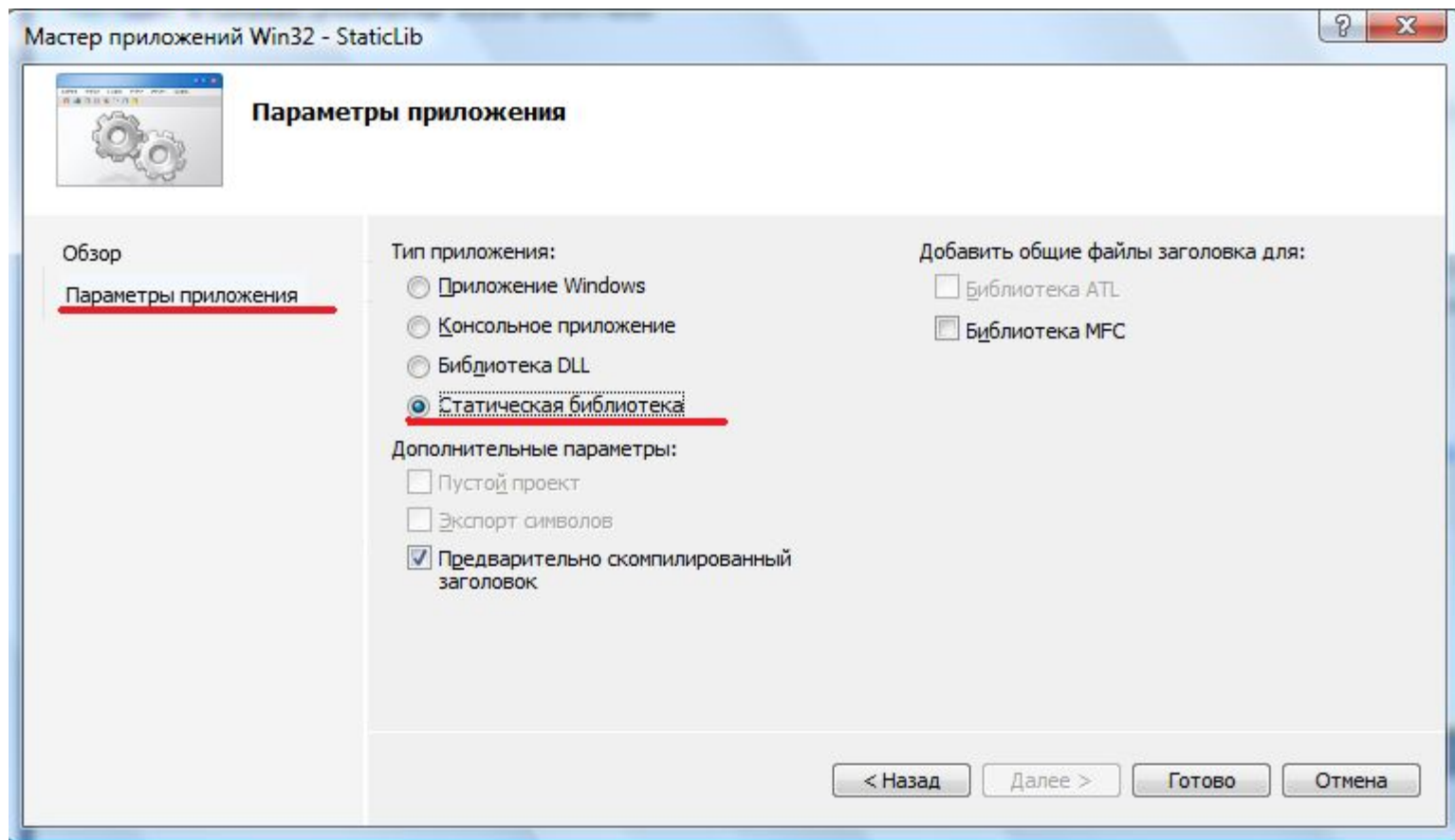
1. Создать проект для статической библиотеки:
File->New->Projects-> Visual C++ -> Win32 -> Win32 Console Application;
2. в ApplicationSettings выбрать Static library;
3. в каталог проекта «исходные файлы» добавить один или несколько файлов, содержащих реализации функций библиотеки;
4. в каталог проекта «заголовочные файлы» добавить заголовочный файл для библиотеки;
5. выбрать соглашение по вызову для функций;
6. создать библиотеку (построить проект) (файл.lib)

Создание статических библиотек.

Шаг 1

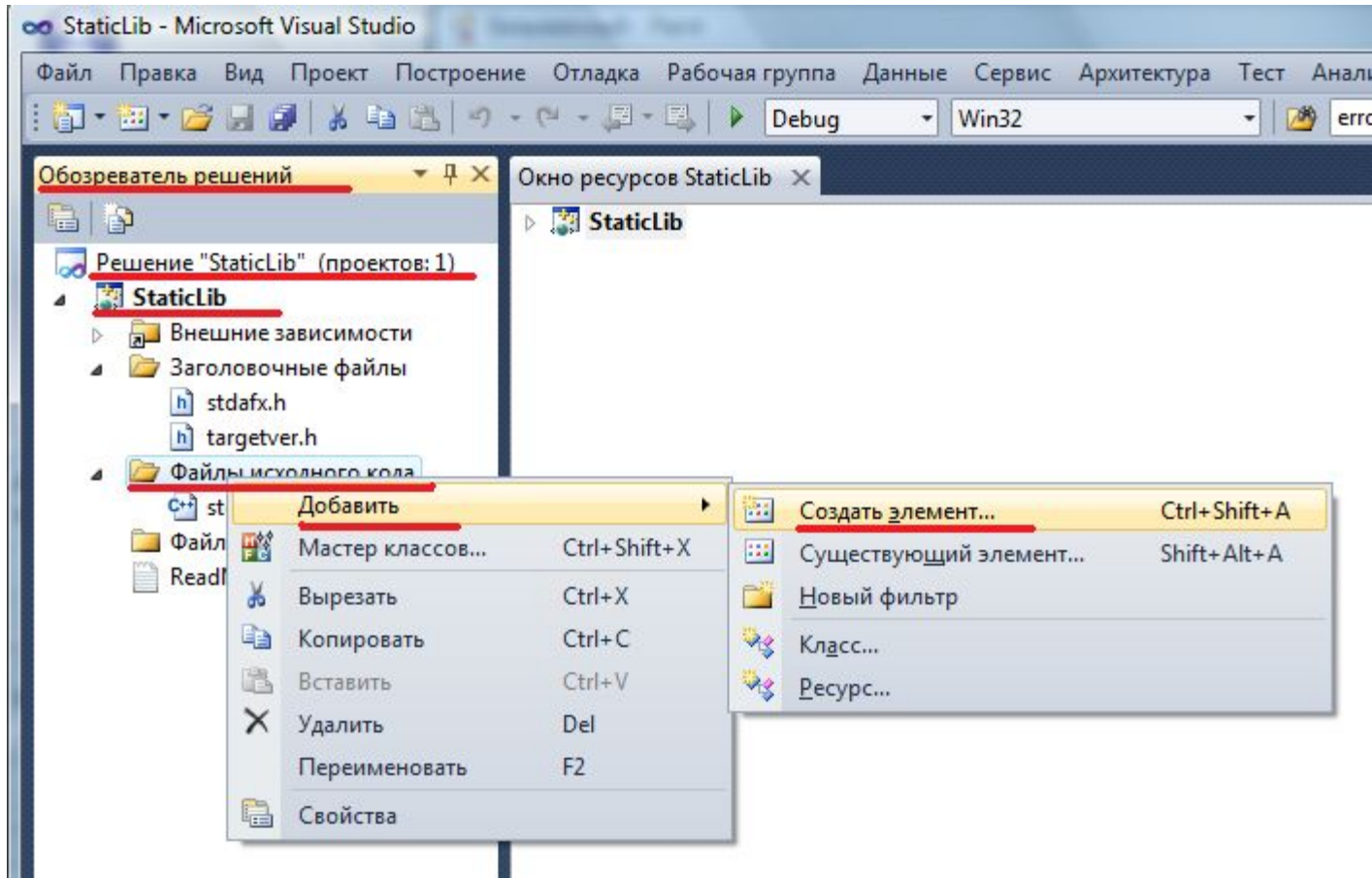


Создание статических библиотек. Шаг 2



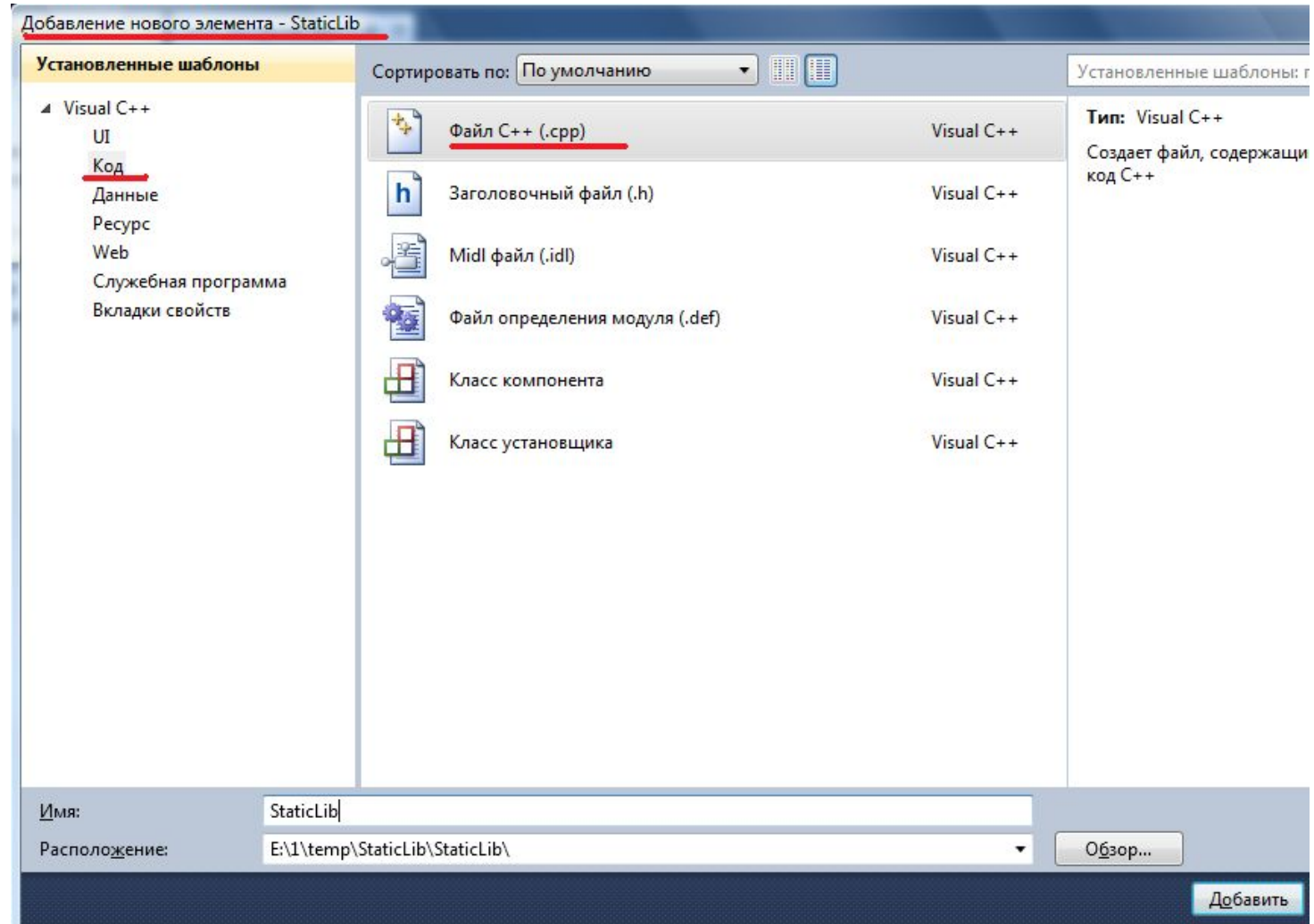
Создание статических библиотек.

Шаг 3



Создание статических библиотек.

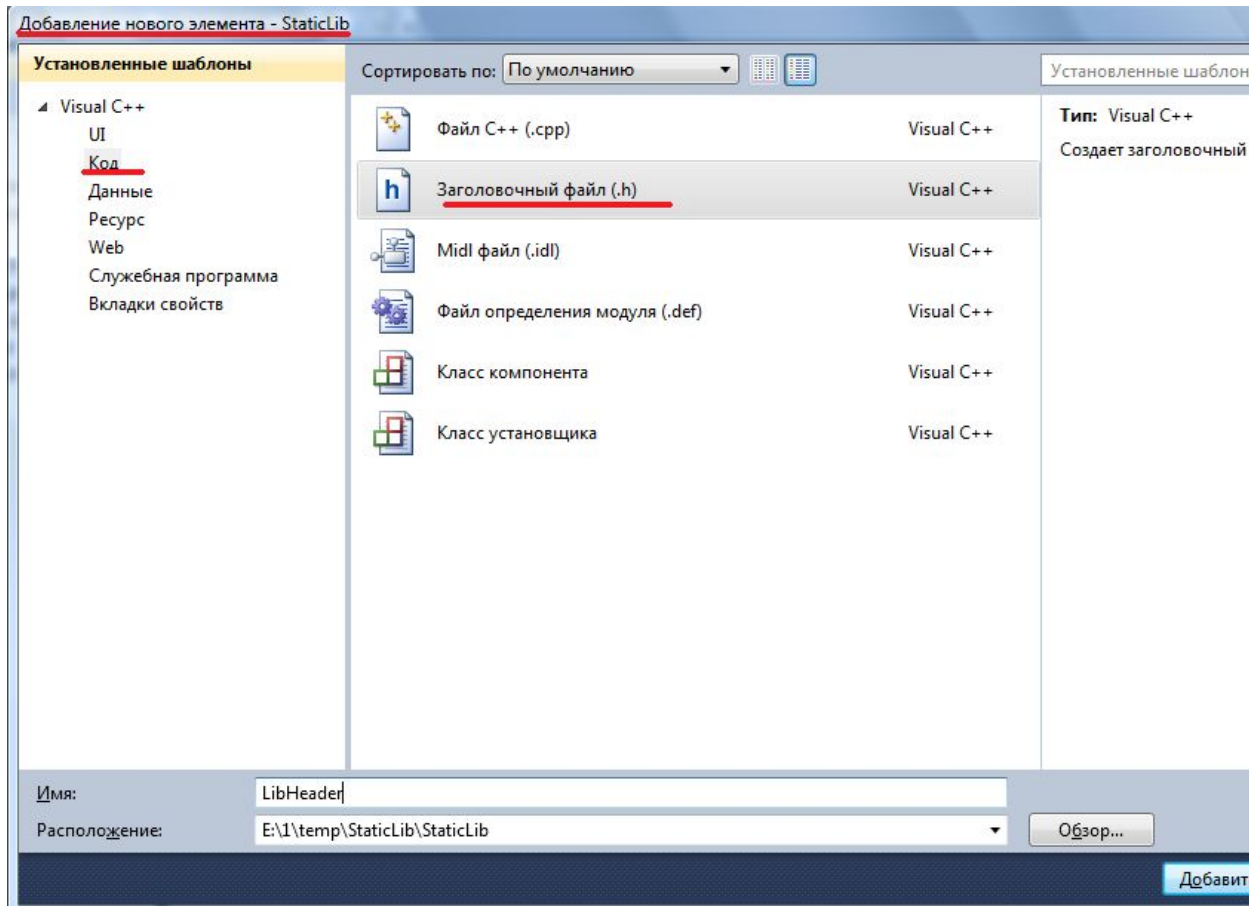
Шаг 3



Создание статических библиотек.

Шаг 4

в каталог заголовочных файлов добавить заголовочный файл для библиотеки



Заголовочные файлы. Правила создания

В нем задают:

- заголовки всех функций, которые входят в библиотеку (интерфейс библиотеки); (например: int summa (int *x, int n);)
- типы пользователя; (например: typedef TCHAR char;)
- константы. (например: #define MAXSIZE 4096)

В нем не задают:

- определение функций, например :

```
int summa (int *x, int n){  
    int s;  
    for (int i = 0; i < n; ++i)  
        s+= x[i];  
    return s;  
}
```

Заголовочные файлы. Обеспечение однократной трансляции

Необходимо обеспечить трансляцию файла только один раз (в противном случае будет ошибка при повторном определении типа или константы с тем же именем).

```
#ifndef    ПРОВЕРКА_КОНСТАНТЫ
#define    ОПРЕДЕЛЕНИЕ_КОНСТАНТЫ
...
#endif
```

Пример статической библиотеки

Разработать библиотеку для выполнения арифметических операций: сложения с учетом возможного переполнения (+) и вычисления НОД для 32 битных чисел.

Требования:

- числа беззнаковые;
- при поиске НОД используется алгоритм вычитания меньшего числа из большего;

Алгоритмы

$$z = x + y;$$

Как определить был перенос или нет?

	7	1	x	f
+ 8		+ 7	+ y	+ f
-----		-----	-----	-----
5		8	z	e

Carry = 1, если $z < x$ или $z < y$;

Наибольший общий делитель : НОД(x,y)

$$x=10, y=15$$

$$15-10=5$$

$$10-5=5$$

$$5-\underline{5}=0$$

$$\text{НОД}(10,15)=5$$

Заголовочный файл (LibHeader.h)

```
// LibHeader.h

#ifndef _LIB_HEADER_H

#define _LIB_HEADER_H

    unsigned int __stdcall AddWithCarry( unsigned int , unsigned int, unsigned int*);

    void __stdcall  NOD( unsigned int a, unsigned int b, unsigned int* r);

#endif
```

Реализация функций (StaticLib.cpp)

```
// StaticLib.cpp

#include "stdafx.h"
#include "LibHeader.h"

unsigned int __stdcall AddWithCarry( unsigned int a, unsigned int b, unsigned int* r)
{
    unsigned int carry=0;
    unsigned int c=a+b;
    if (c<a)
        carry=1;
    *r=c;
    return carry;
}
```

(продолжение на следующем слайде)

Реализация функций (StaticLib.cpp)

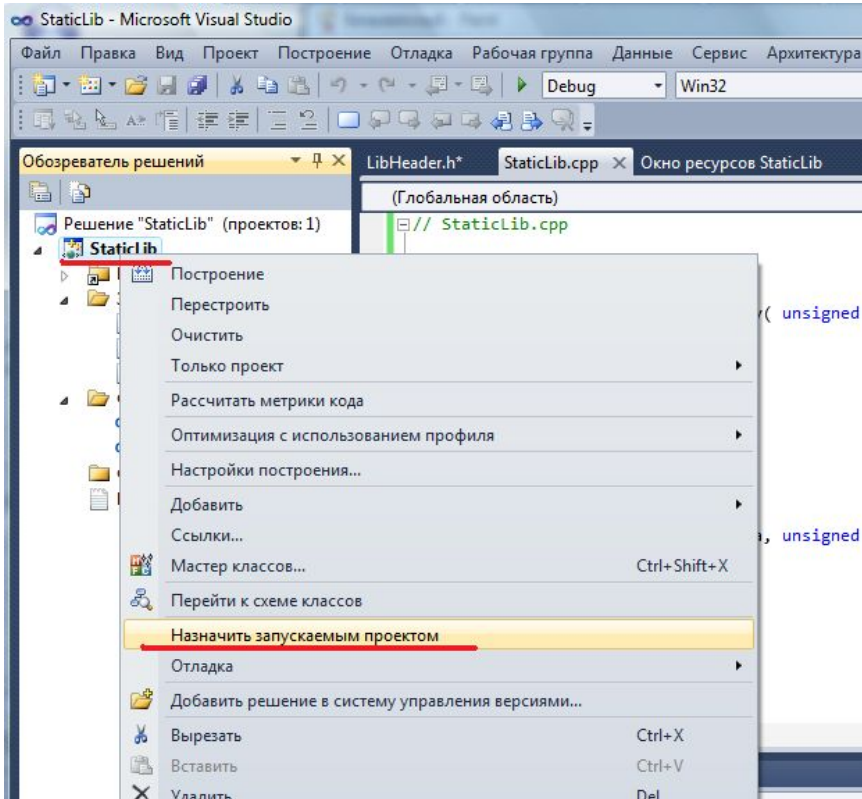
(Продолжение. Начало на предыдущем на следующем слайде)

```
void __stdcall NOD( unsigned int a, unsigned int b, unsigned int* r)
{
    unsigned int f=a, s=b;

    while( f&& s )
    {
        if ( f>s )
            f=f-s;
        else
            s=s-f;
    }

    if (f)
        *r=f;
    else
        *r=s;
}
```

Построение статической библиотеки.



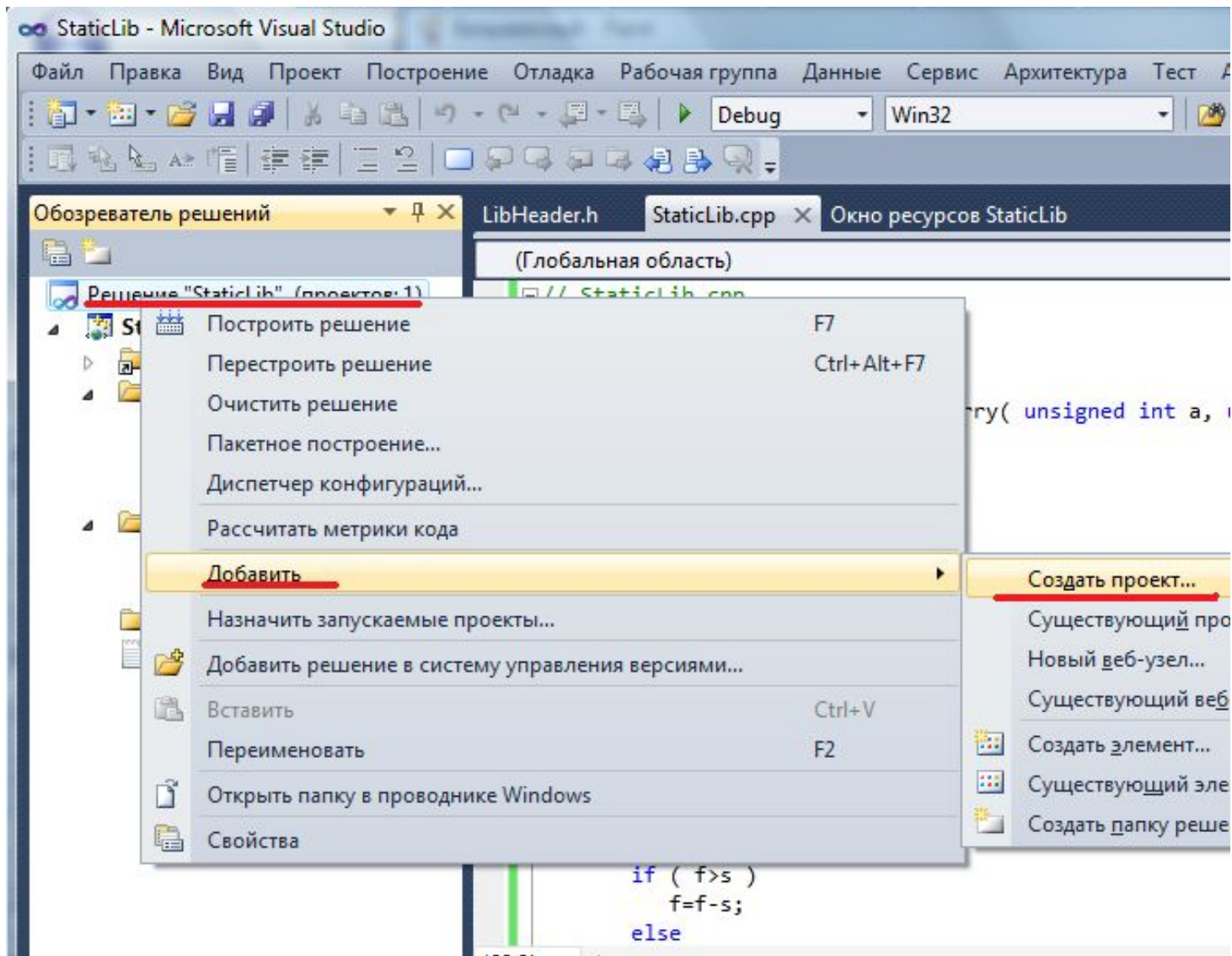
если в Решении (Solution) несколько проектов - выбрать пункт контекстного меню Set As StartUp Project;

далее
выбрать пункт меню Построение (Build)

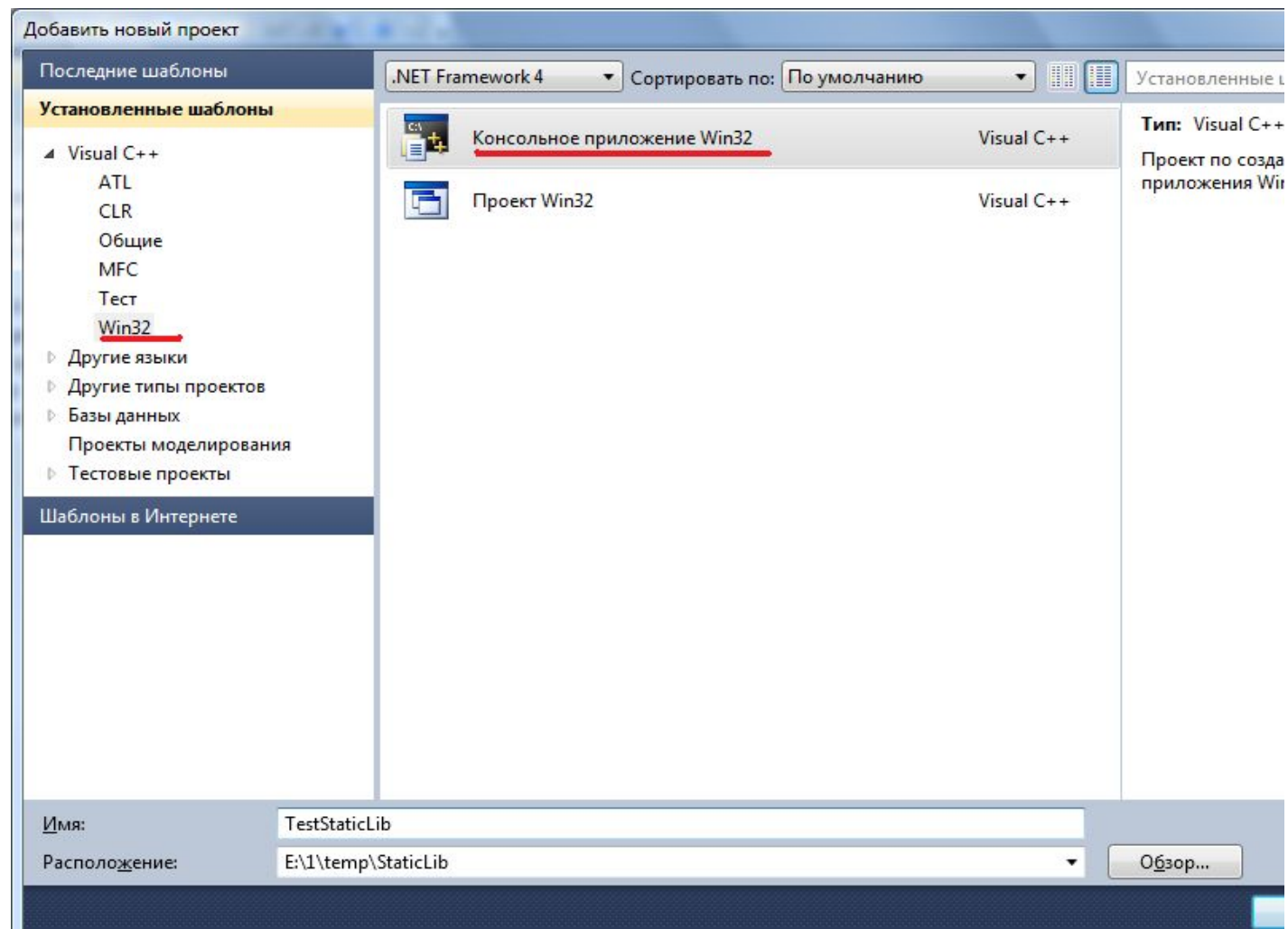
В результате построения проекта Static Library получим файл .lib в каталоге Debug Вашего Решения (Solution)

(если текущая конфигурация - Debug)

Главная программа для использования статической библиотеки. Создание



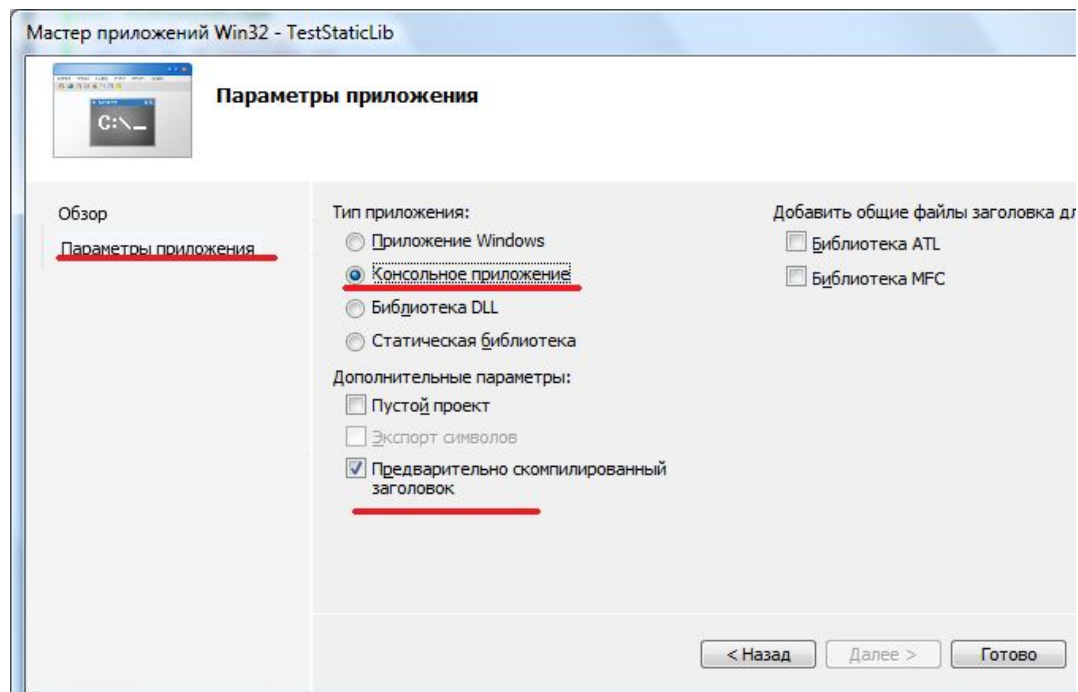
Главная программа для использования статической библиотеки. Создание



Главная программа для использования статической библиотеки. Создание

- в ApplicationSettings оставить Console Application и Precompiled header;
- Откроется исходный файл созданного проекта со следующим содержанием

```
#include "stdafx.h"  
  
int _tmain(int argc, _TCHAR* argv[])  
{  
    return 0;  
}
```



Главная программа для использования статической библиотеки. Настройки.

Использовать контекстное меню

или

выбрать

пункт главного меню

Project-> Properties ->

Configuration Properties->

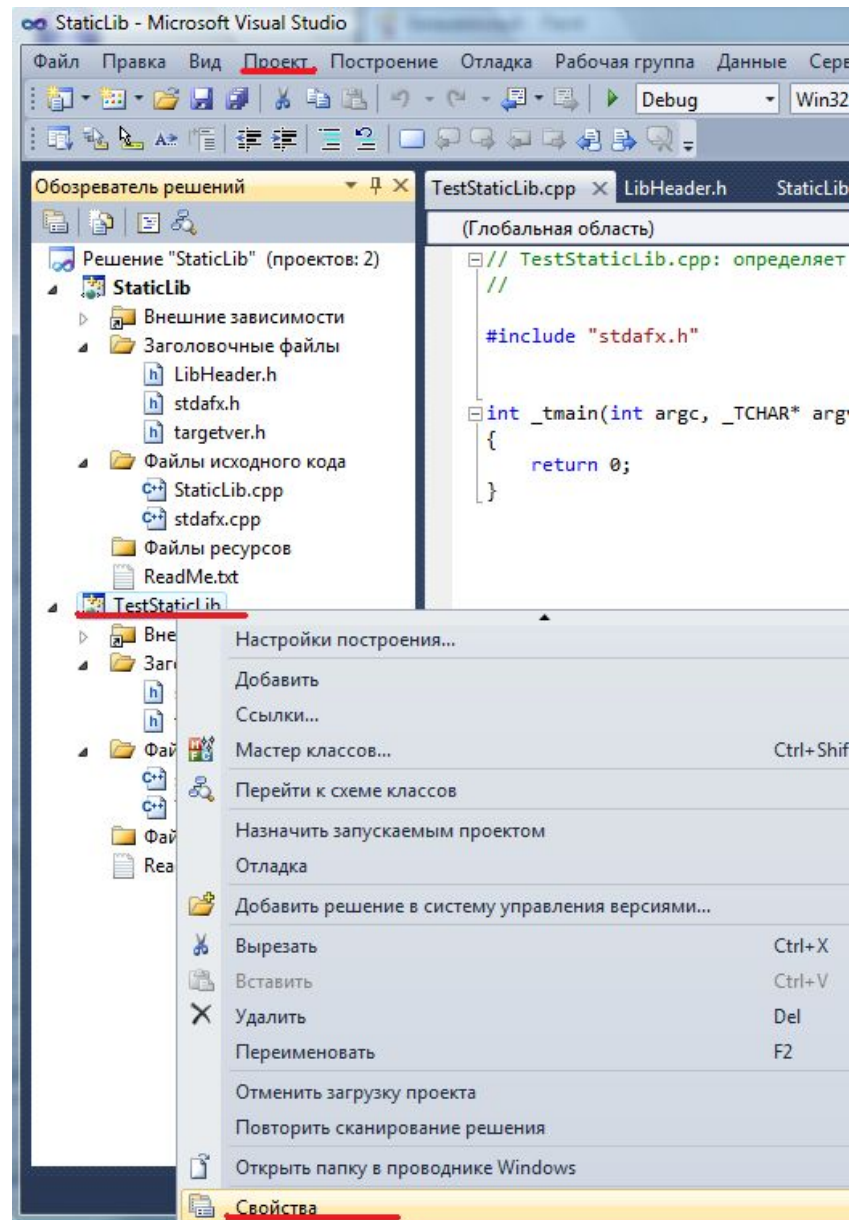
C/C++->General->

Additional Include Directories

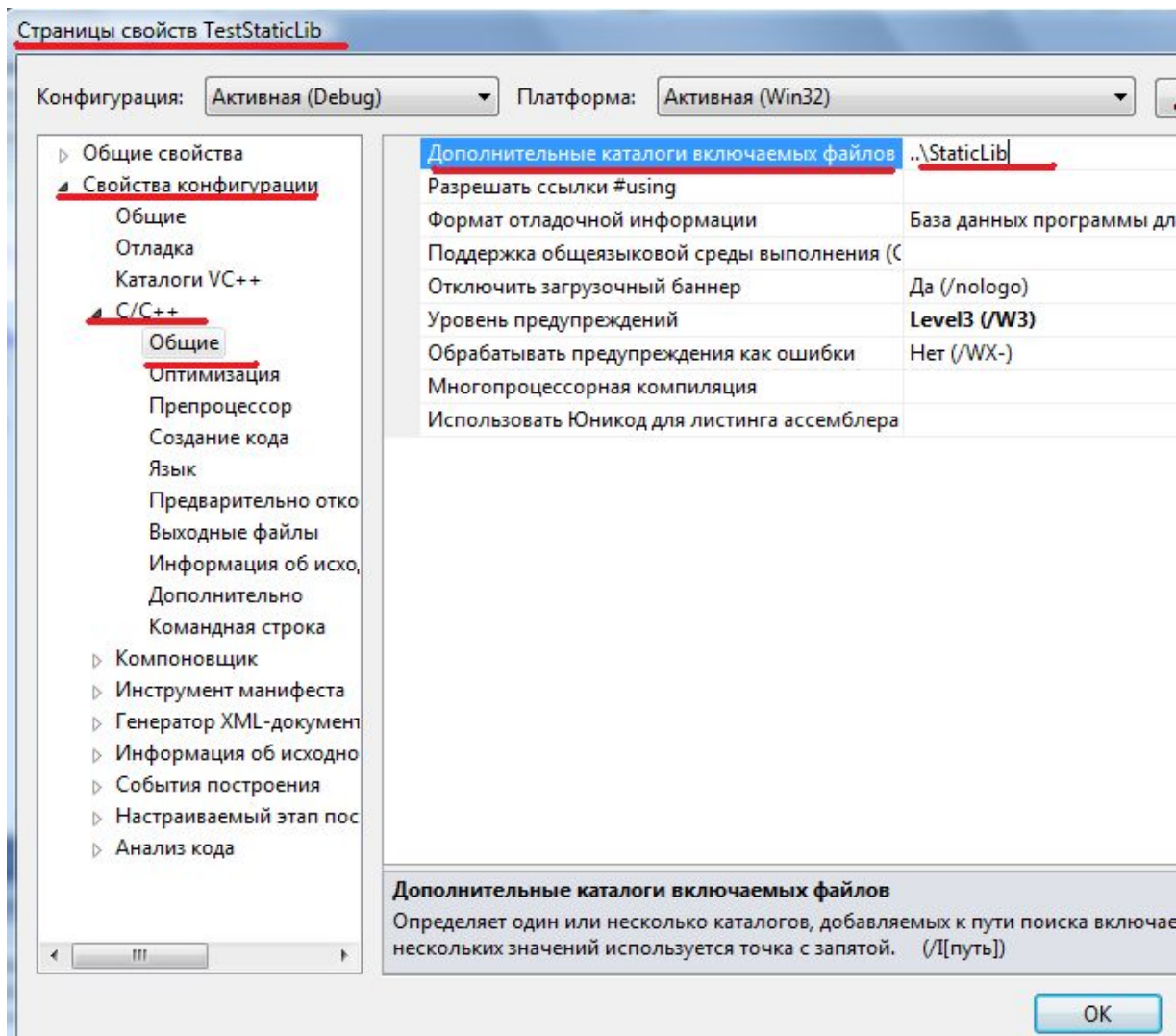
установить как

..\ИМЯ_ПРОЕКТА_СТАТИЧЕСКАЯ_БИБЛ;

(заголовочный файл один на все проекты
в Решении (Solution))



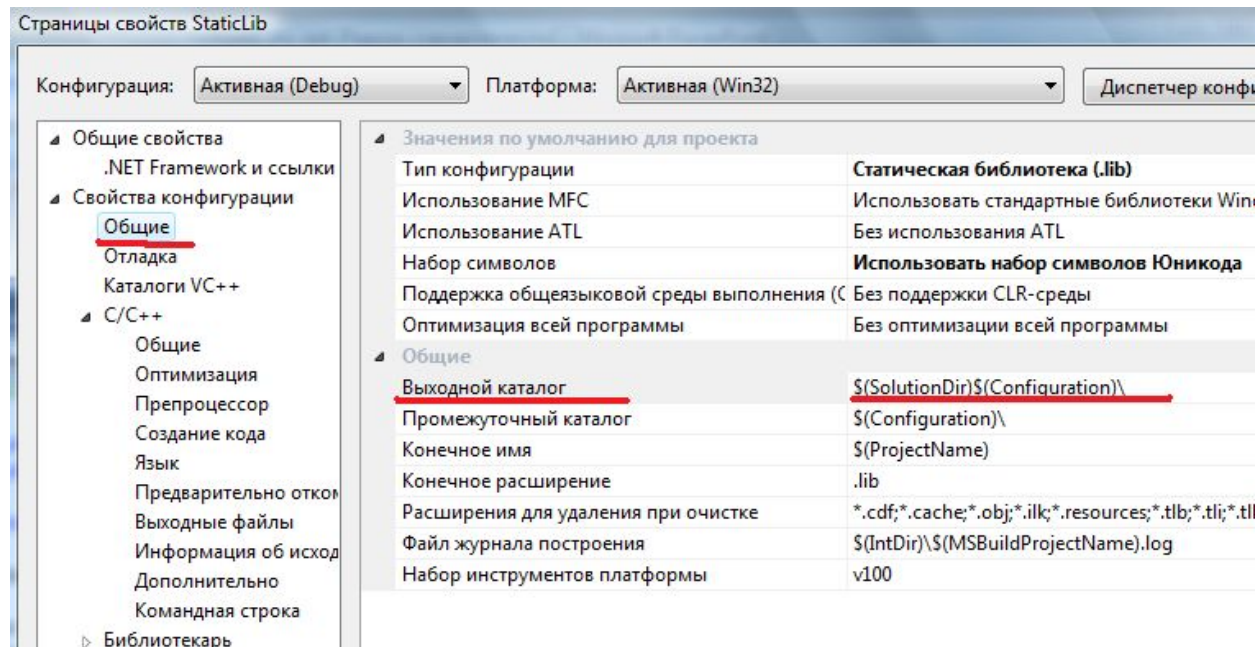
Главная программа для использования статической библиотеки. Настройки.



Использование статических библиотек.

Рекомендации

- В одном Решении (Solution) создать 2 проекта: проект для библиотеки и проект с главной программой.
- для обоих проектов резльтирующий каталог должен быть один и тот же меню Project-> Properties -> Configuration Properties-> General-> Output Directory -> **$\$(SolutionDir)\$(ConfigurationName)\$**



Главная программа TestStaticLib.cpp. (для вызова функций из библиотеки StaticLib)

```
#include "stdafx.h"
#include "LibHeader.h"

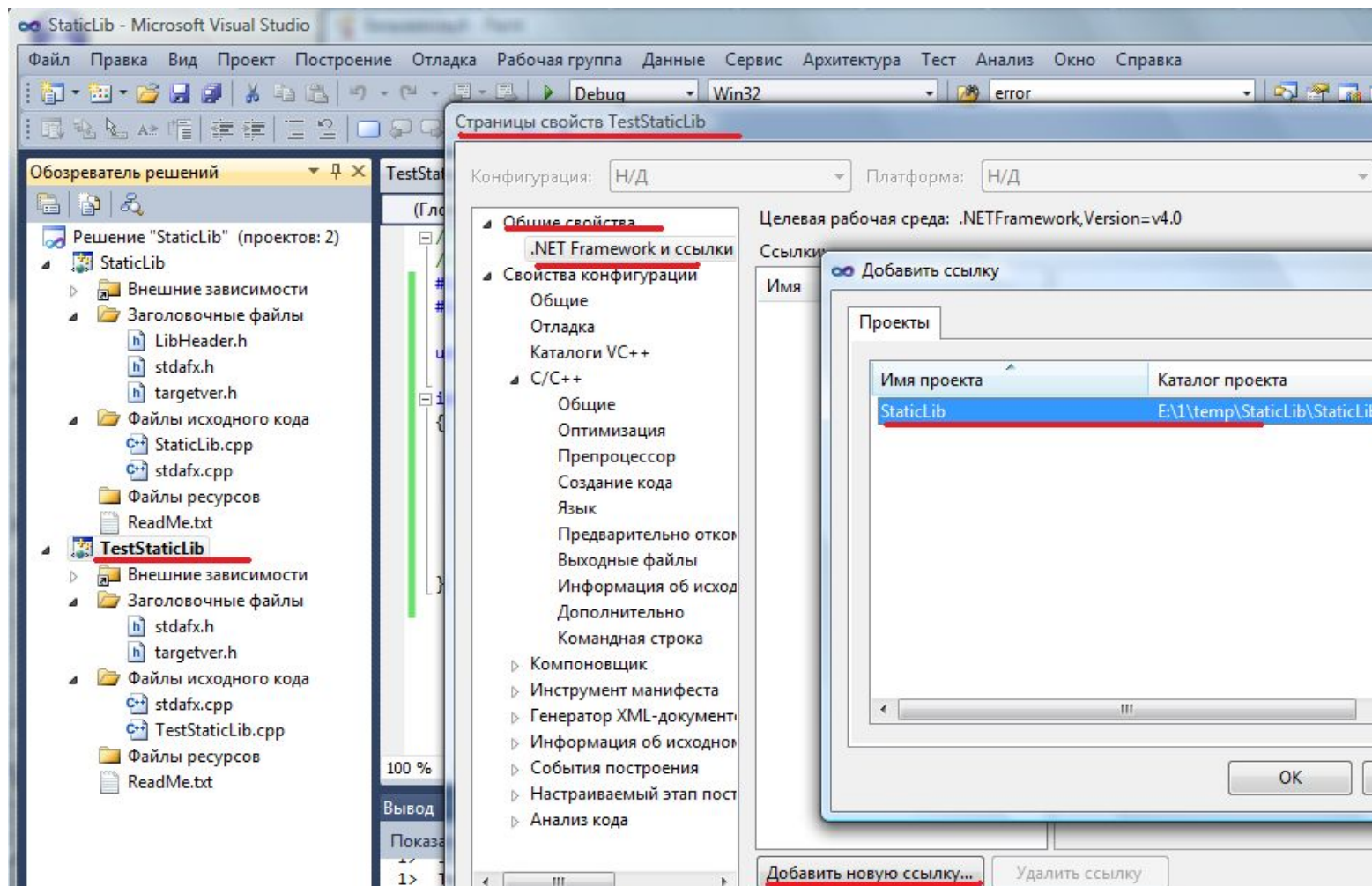
unsigned int first=0xffffffff, second=0xffffffff, result, carry;

int _tmain(int argc, _TCHAR* argv[])
{
    carry=AddWithCarry(first, second, &result);
    _tprintf(_T("%x+%x= %x %x\n"), first, second, carry, result);

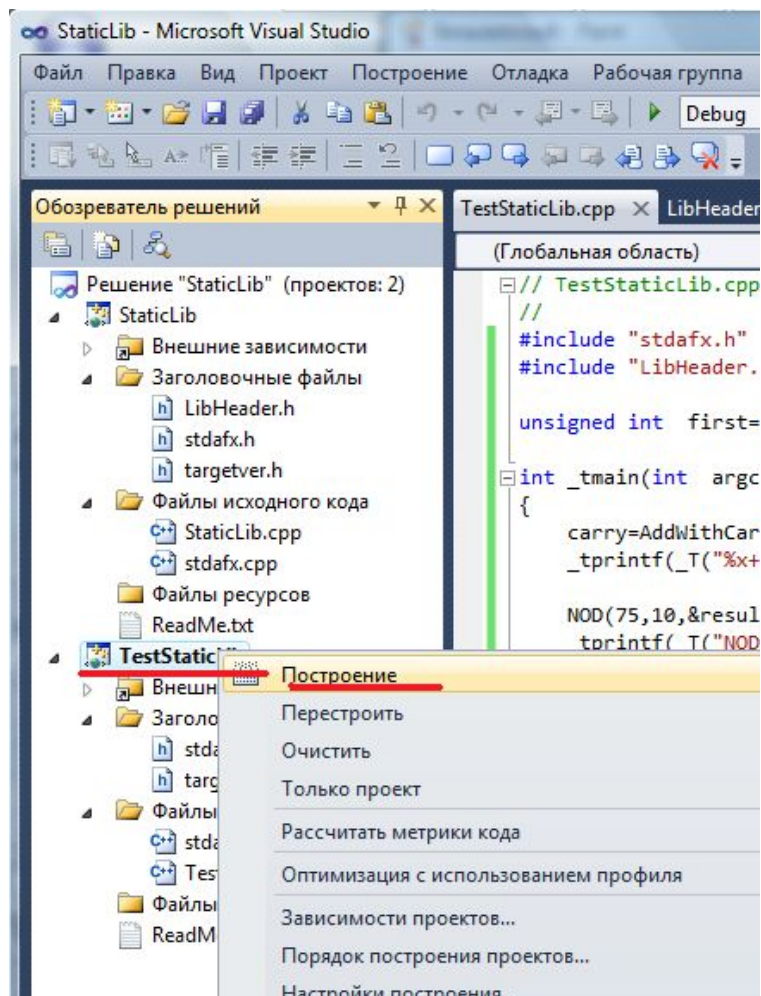
    NOD(75,10,&result);
    _tprintf(_T("NOD for %u and %u = %u"),75,10,result);
    return 0;
}
```

Главная программа для использования статической библиотеки. Ссылка на библиотеку.

Пункт меню Project->References.. ->Add New Reference..->выбрать имя проекта с библи.



Главная программа для использования статической библиотеки . Построение и запуск программы.



выбрать пункт меню Set As StartUp Project;

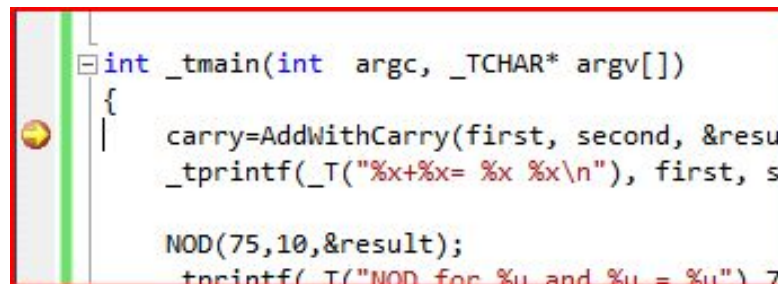
далее

выбрать пункт меню Build ->
Build TestStaticLib или воспользоваться
КОНТЕКСТНЫМ МЕНЮ.

далее

выбрать пункт главного меню Debug ->
Start Without Debugging (Ctrl+F5)
или **Start Debugging (F5)**

**Поставить точку останова и в пошаговом
режиме отладить программу (F10, F11)**



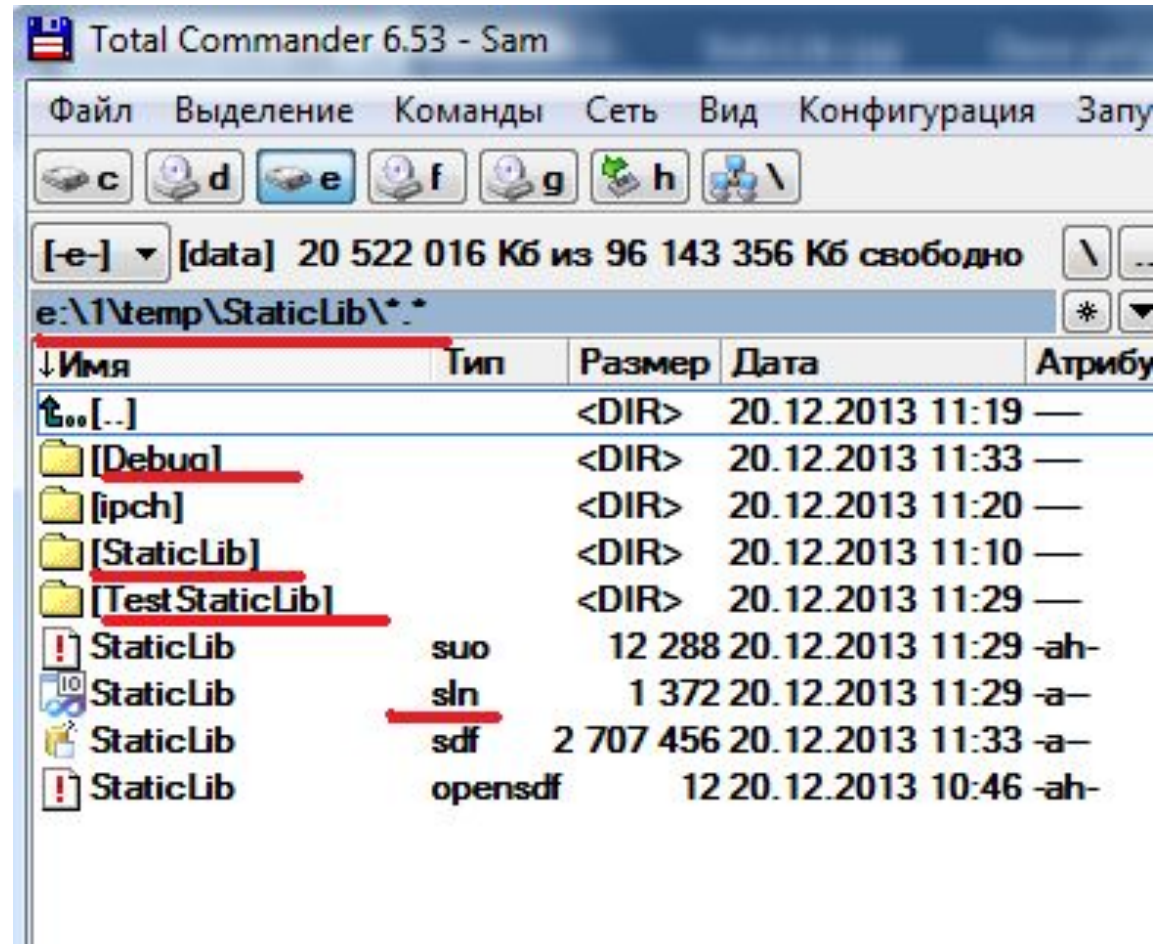
Каталоги с Решением (solution), содержащим оба проекта.

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

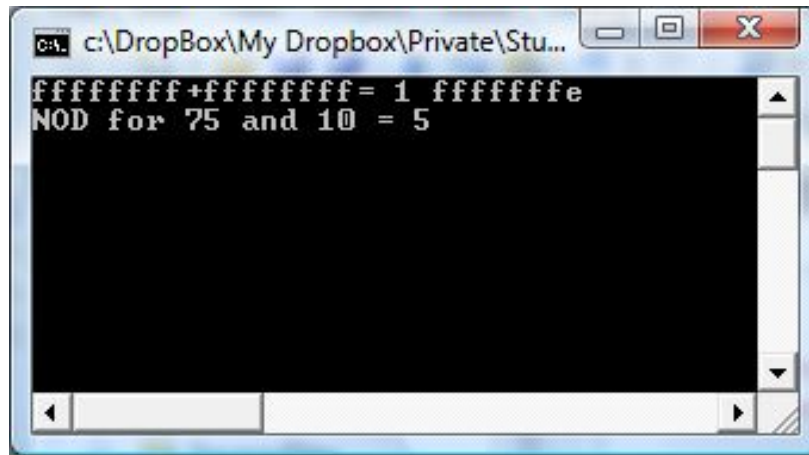
Имя Решения:
StaticLib

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.



```
c:\DropBox\My Dropbox\Private\Stu...
ffffffff+ffffffff= 1 fffffffe
NOD for 75 and 10 = 5
```

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:

- платформенно зависима;
- 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.

План лекции №4

- Создание динамических библиотек;
- Использование динамических библиотек. Статический режим;
- Использование динамических библиотек. Динамический режим;
- Пример создания и использования;
- Преимущества и недостатки разных режимов использования.

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)`

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!

(пример: страны экспортеры и импортеры)

```
// 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 функции2
```

```
...
```

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
#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

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. Построить проект. В результате будут получены 2 файла: <имя>.lib и <имя>.dll;

Использование:

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

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 необходимости время выполнения приложения, которое использует DLL) - загружаются только те DLL, из которых будут вызываться функции, после окончания использования память можно освободить, не дожидаясь окончания работы главной программы.

DEF файл

Добавляется в проект DLL (в папку ресурсов) для сохранения возможности обращения к функциям по именам без преобразования при использовании DLL в режиме динамической загрузки.

EXPORTS

Имя функции 1

Имя функции 2

...

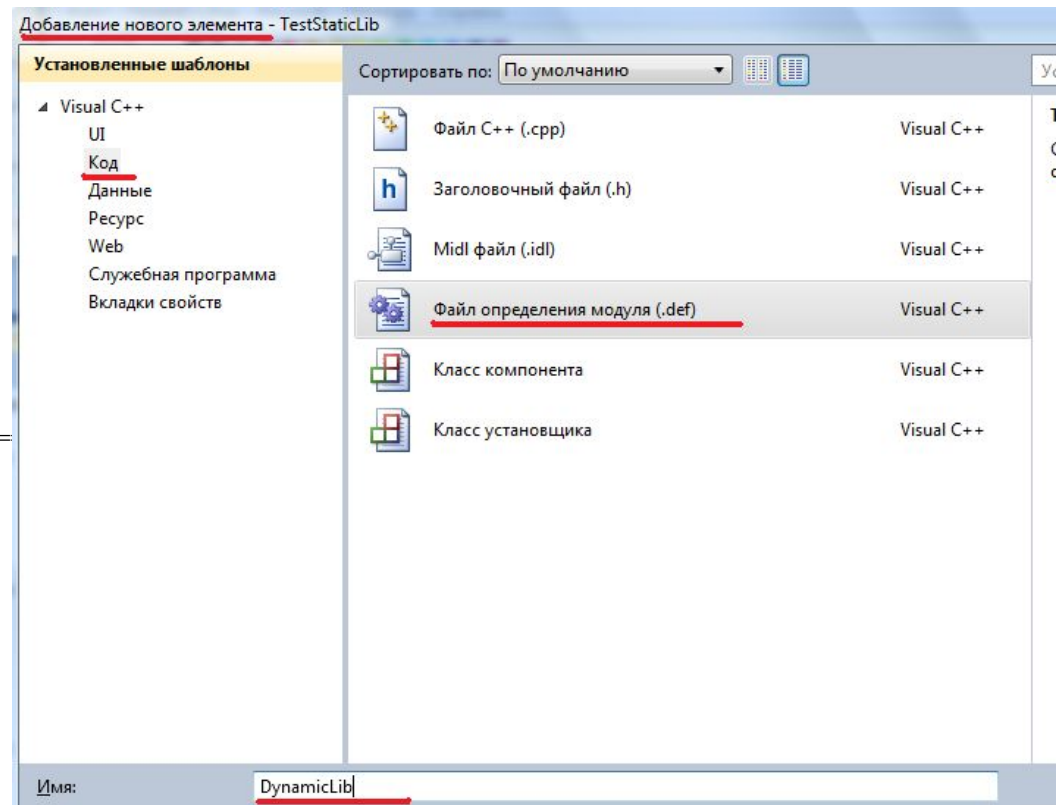
DynamicLib.def:

LIBRARY "DynamicLib"

EXPORTS

AddWithCarry

NOD



Правила использования функций WIN 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*);
- DWORD WINAPI GetDllDirectory(DWORD *nBufferLength*, LPTSTR *lpBuffer*);

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;
    HINSTANCE 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("%0x+%0x= %0x %0x\n"),first,second,carry,result);

        adr2(75,10,&result);
        _tprintf(_T("NOD for %u and %u = %u"),75,10,result);
    }
    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 проекту ДЛЛ