

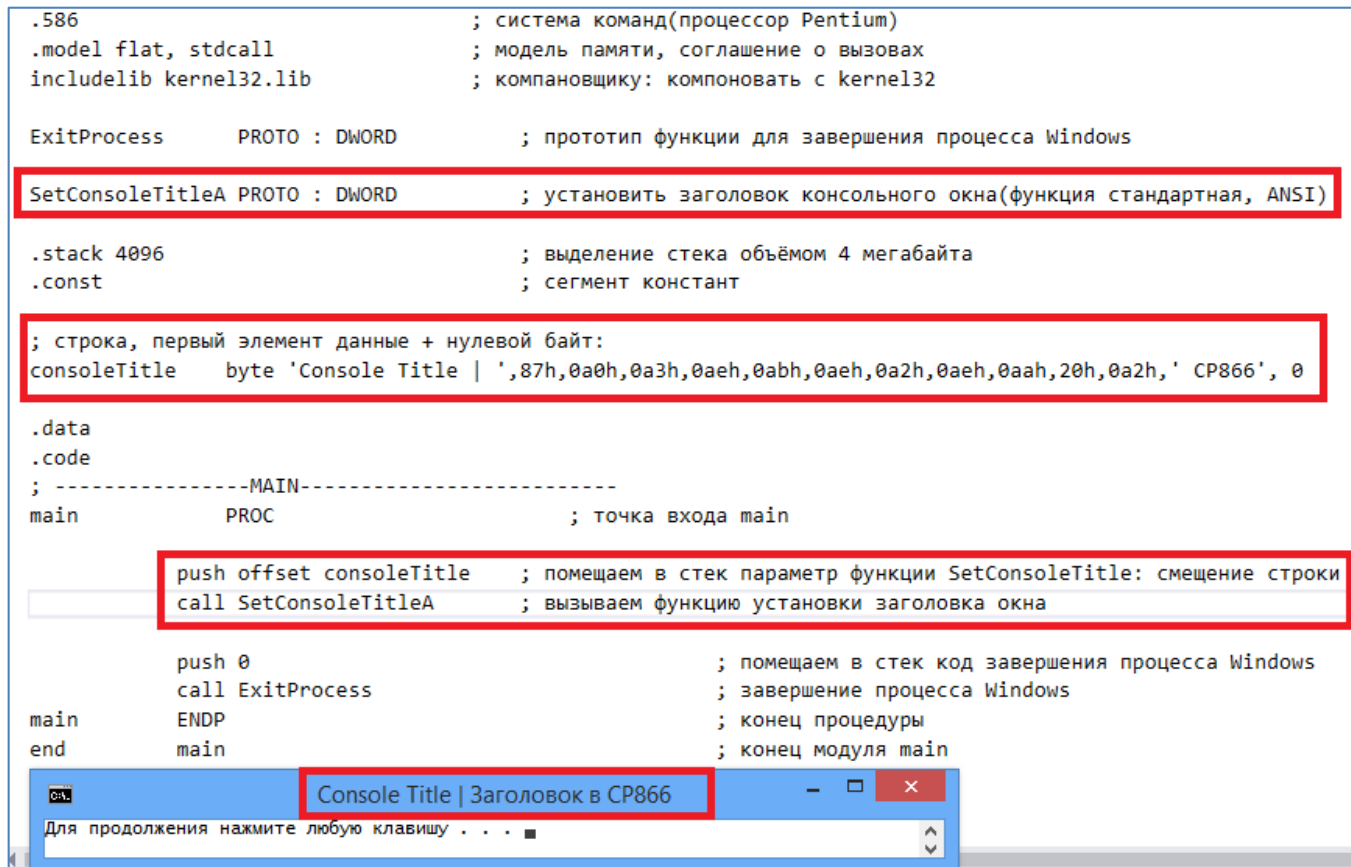
# БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования

## Введение в язык Ассемблер

**Тема: Определение и использование процедур. Статическая библиотека.**

### 1. Вывод в консоль

#### 1.1 Установить заголовок для консольного окна



#### Синтаксис:

```

BOOL SetConsoleTitle(    // устанавливает заголовок для консольного окна
    LPCTSTR lpConsoleTitle // 32-разрядный указатель на строку lpConsoleTitle –
                           // заголовок консольного окна
);
  
```

**Необходимая библиотека:** Kernel32.lib

**Соглашение о вызовах:** stdcall

**Возвращаемые значения:** { 0 – функция завершается с ошибкой;  
 иначе – функция завершается успешно.

## 1.2 Вывод в консоль

```
ExitProcess    PROTO : DWORD          ; прототип функции для завершения процесса Windows
SetConsoleTitleA PROTO : DWORD          ; установить заголовок консольного окна(функция стандартная, ANSI)
GetStdHandle   PROTO : DWORD          ; получить handle вывода на консоль
              ; (принимает константное значение - 10 ввод, -11 вывод, -12 ошибка устройства вывода)
WriteConsoleA  PROTO : DWORD, : DWORD, : DWORD, : DWORD, : DWORD ; вывод на консоль(стандартная функция)

.stack 4096          ; выделение стека объёмом 4 мегабайта
.const             ; сегмент констант
.data
consoleTitle  byte 'Console Title | ',87h,0a0h,0a3h,0aeh,0abh,0aeh,0a2h,0aeh,0aah,20h,0a2h,' CP866', 0
helloworld    byte "Hello World!!!"
              byte endl
HW             = ($ - helloworld)      ; вычисление длины строки helloworld
messageSize   dword ?
consolehandle  dword 0h                ; состояние консоли

.code
; -----MAIN-----
main          PROC                    ; точка входа main
              push offset consoleTitle ; в стек 1-й параметр функции SetConsoleTitle: смещение строки
              call SetConsoleTitleA    ; вызов функции установки заголовка окна SetConsoleTitleA
; -----
              mov messageSize, HW
              push -11                  ; -11 - handle для стандартного вывода
              call GetStdHandle         ; получить handle -> eax
              mov consolehandle, eax   ; сохраняем его в consolehandle
; -----
              push 0                    ; можно 0 (резерв)
              push 0                    ; можно 0
              push messageSize          ; количество байт
              push offset helloworld    ; адрес выводимой строки
              push consolehandle        ; handle для вывода
              call WriteConsoleA        ; вывести в консоль
```



### Синтаксис:

```
HANDLE GetStdHandle(          // извлекает дескриптор потока ввода-вывода
    DWORD nStdHandle          // ввод, вывод или ошибка
);
```

Handle стандартного потока **ввода** -10  
 Handle стандартного потока **вывода** -11  
 Handle потока сообщений об ошибках **"ошибок"** -12

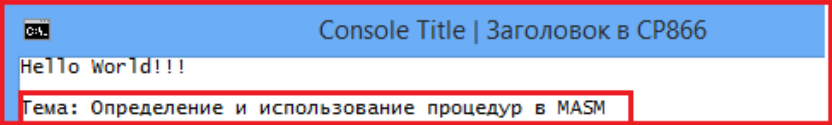
### Синтаксис:

```
BOOL WriteConsole(            // выводит символьную строку в консоль
    HANDLE hConsoleOutput,    // дескриптор (Handle)
    CONST VOID * lpBuffer,     // указатель на строку вывода
    DWORD nNumberOfCharsToWrite, // число выводимых символов
    LPDWORD lpNumberOfCharsWritten, // возвращает число выведенных символов
    LPVOID lpReserved          // зарезервировано
);
```

## 2. Процедура вывода

### 2.1 Вызов процедуры printconsole

```
.586                                ; система команд(процессор Pentium)
.model flat, stdcall                ; модель памяти, соглашение о вызовах
includelib kernel32.lib             ; компоновщику: компоновать с kernel32
includelib ucrt.lib                 ; библиотека времени исполнения C
;-----
system      PROTO C : DWORD          ; вызов cmd команды
;-----
printconsole PROTO : DWORD, : DWORD  ; вызов поцедуры вывода в консоль
;-----
SetConsoleOutputCP PROTO : DWORD     ; устанавливает номер входной кодовой страницы для терминала
SetConsoleCP       PROTO : DWORD     ; устанавливает номер выходной кодовой страницы для терминала
.stack 4096          ; выделение стека объёмом 4 мегабайта
.const              ; сегмент констант
endl               equ 0ah           ; символ перевода строки (ASCII)
str_endl          byte endl,0       ; строка "конец строки"
;-----
.data
begin            byte "Тема: Определение и использование процедур в MASM",0
str_pause       byte "pause", 0     ;
;-----
push 1251d
call SetConsoleOutputCP
push 1251d
call SetConsoleCP
invoke printconsole, offset begin, offset consoleTitle ; вывод конца строки
;-----
invoke printconsole, offset str_endl, offset consoleTitle ; вывод конца строки
```



## 2.2 Процедура вывода сообщений в консоль printconsole

```
; -----printconsole-----
printconsole  proc uses eax ebx ecx edi esi,
               pstr :dword,
               ptitle :dword

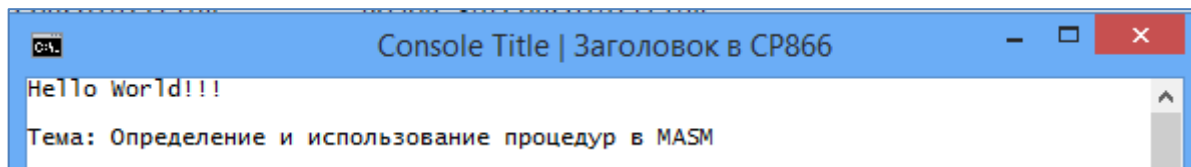
               push ptitle          ; параметр SetConsoleTitleA: адрес строки заголовка
               call SetConsoleTitleA ; вызов SetConsoleTitleA
               push -11             ; -11 - handle для стандартного вывода
               call GetStdHandle     ; получить handle -> в eax

count:        mov esi, pstr         ; подсчет количества символов
               mov edi, -1          ; до 0-символа
               ; выводимой
               inc edi              ; на консоль строке
               cmp byte ptr [esi + edi], 0 ;
               jne count            ; если не 0-символ, на метку count

               push 0               ; можно 0 (резерв)
               push 0               ; можно 0
               push edi             ; количество байт
               push pstr            ; адрес выводимой строки
               push eax             ; handle для вывода (eax)
               call WriteConsoleA   ; вывести в консоль

               ret

printconsole  ENDP
```



### 3. Процедура преобразования числа в символы

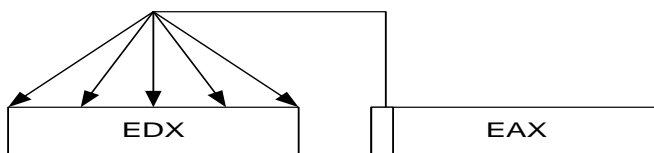
```

; -----преобразование числа в строку-----
int_to_char PROC uses eax ebx ecx edi esi,
    pstr : dword,      ; адрес строки результата
    intfield : sdword  ; число для преобразования

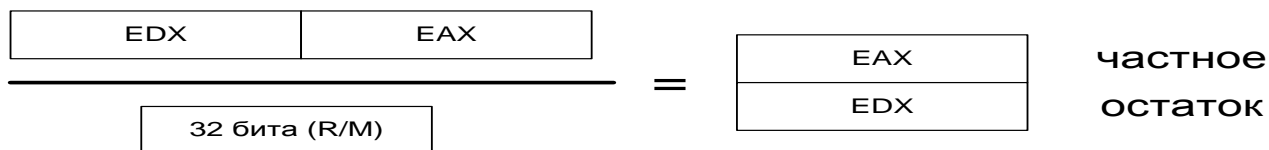
    mov edi, pstr      ; копирует из pstr в edi
    mov esi, 0         ; количество символов в результате
    mov eax, intfield  ; число -> в eax
    cdq               ; знак числа распространяется с eax на edx
    mov ebx, 10        ; основание системы счисления (10) -> ebx
    idiv ebx           ; eax = eax/ebx, остаток в edx (деление целых со знаком)
    test eax, 80000000h ; тестируем знаковый бит
    jz plus           ; если положительное число - на plus
    neg eax            ; иначе меняем знак eax
    neg edx            ; edx = -edx
    mov cl, '-'        ; первый символ результата '-'
    mov[edi], cl       ; первый символ результата '-'
    inc edi            ; ++edi
    plus :             ; цикл разложения по степеням 10
    push dx            ; остаток -> стек
    inc esi            ; ++esi
    test eax, eax      ; eax == ?
    jz fin            ; если да, то на fin
    cdq               ; знак распространяется с eax на edx
    idiv ebx           ; eax = eax/ebx, остаток в edx
    jmp plus          ; безусловный переход на plus
fin :
    mov ecx, esi       ; в ecx количество не 0-вых остатков = количеству символов результата
write :
    pop bx            ; остаток из стека -> bx
    add bl, '0'        ; сформировали символ в bl
    mov[edi], bl       ; bl -> в результат
    inc edi            ; edi++
    loop write         ; если (--ecx)>0 переход на write
    ret
int_to_char ENDP

```

#### CDQ



#### DIV/IDIV



```

.586                                ; система команд (процессор Pentium)
.model flat, stdcall                ; модель памяти, соглашение о вызовах
includelib kernel32.lib             ; компоновщик: компоновать с kernel32.lib
ExitProcess      PROTO :DWORD       ; прототип функции
includelib msvcrt.lib               ; библиотека времени исполнения C
system           PROTO C :DWORD     ; вывод cmd-команды
.stack 4096                          ; сегмент стека объемом 4096
.const           ; сегмент констант
consoletitle    db 'int_to_char',0
str_pause       db 'pause',0
.data           ; сегмент данных
result1         byte 40 dup(0)
                byte 10
result2         byte 40 dup(0)
.code           ; сегмент кода
main PROC      ; начало процедуры

push -777777777 ; исходное число
push offset result1 ; место для результата
call int_to_char ; вызов процедуры преобразования

push offset consoletitle ; заголовок окна консоли
push offset result1 ; выводимый текст
call printconsole ; вызов процедуры

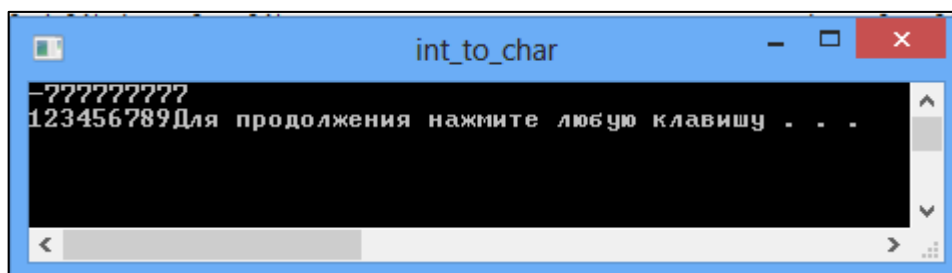
push 123456789 ; исходное число
push offset result2 ; место для результата
call int_to_char ; вызов процедуры преобразования

push offset consoletitle ; заголовок окна консоли
push offset (result2-1) ; выводимый текст
call printconsole ; вызов процедуры

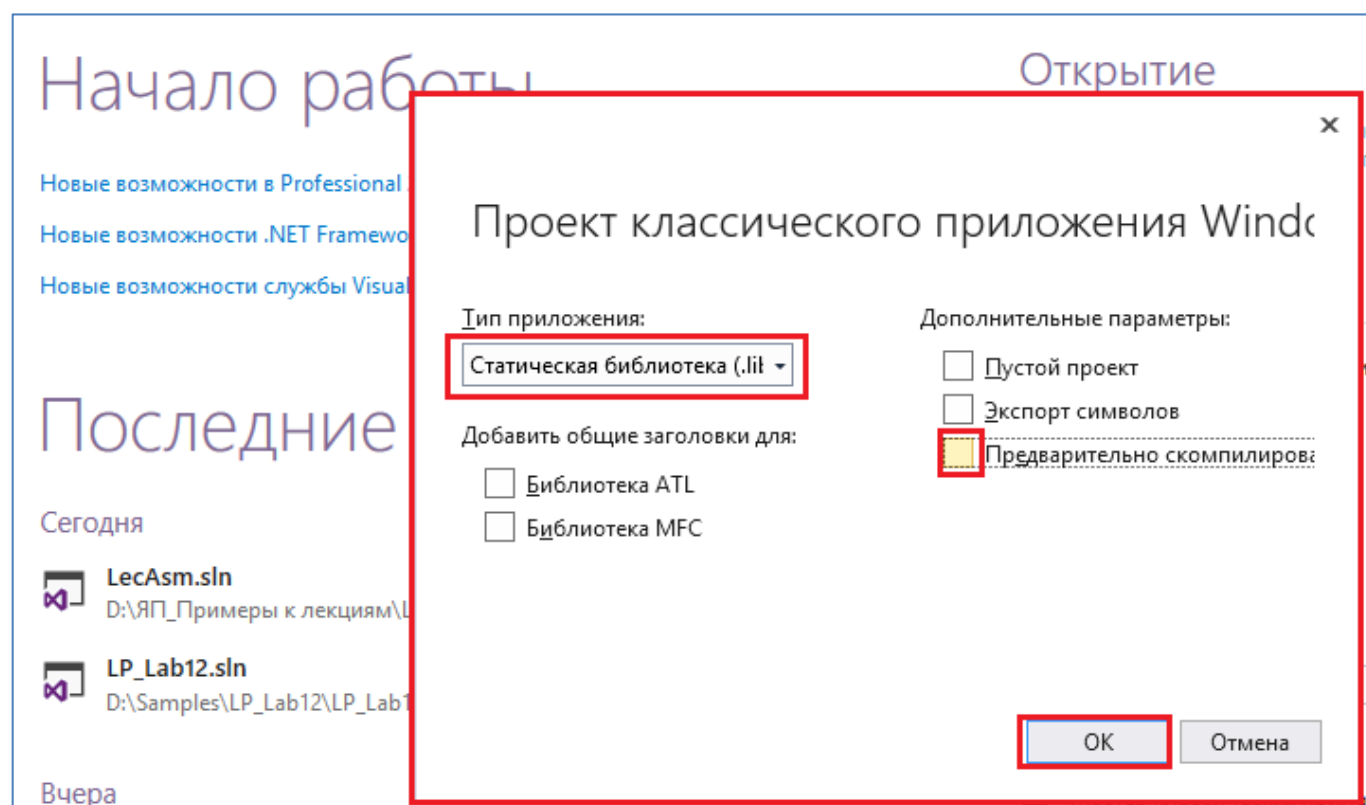
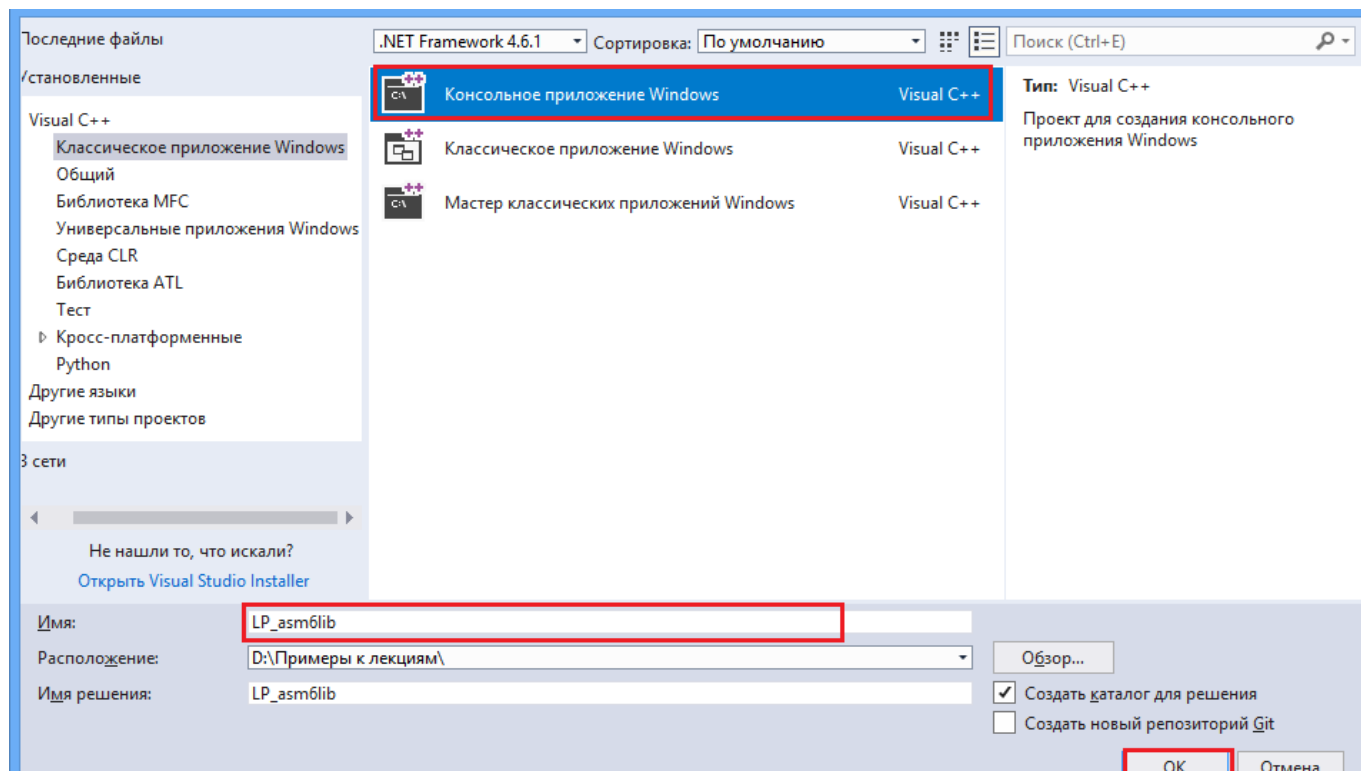
push offset str_pause ; адрес выводимой cmd-команды
call system ; system("pause");

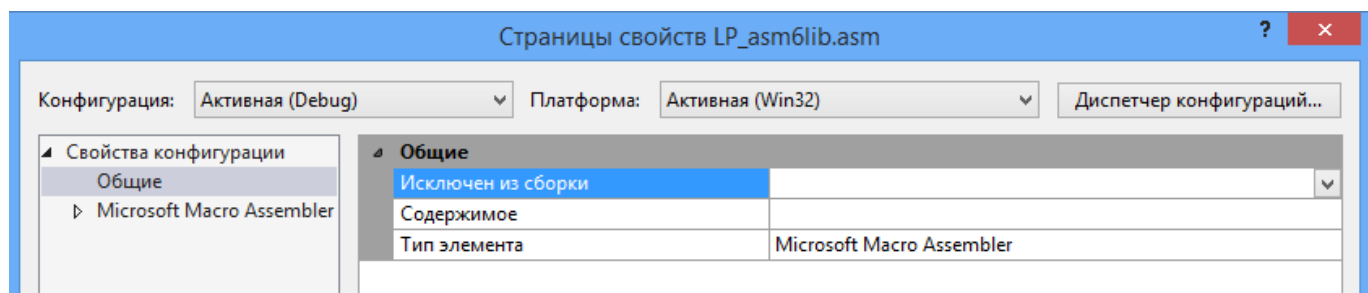
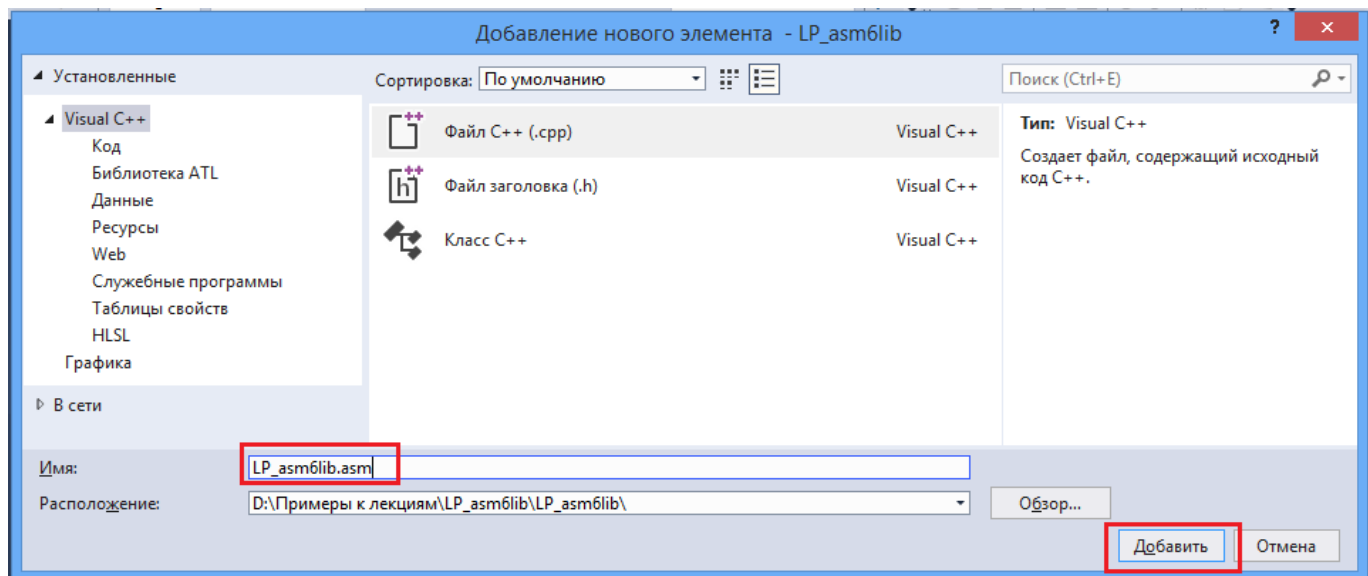
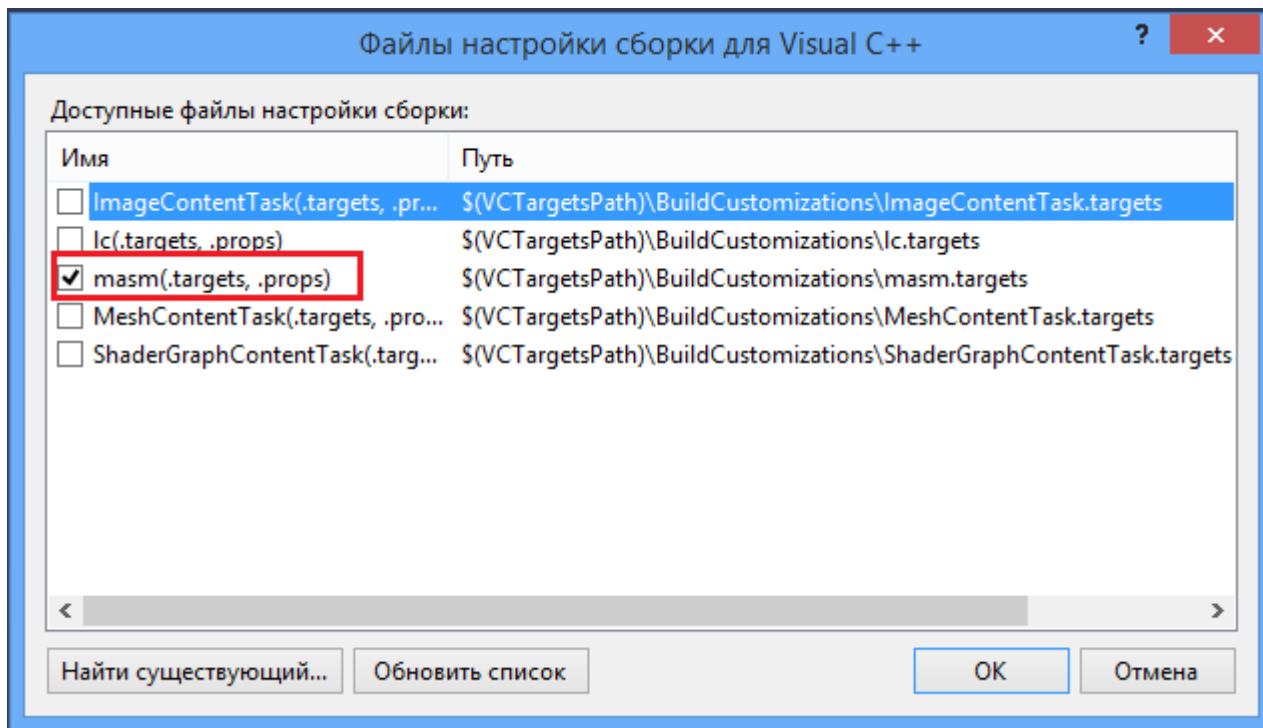
push 0 ; код возврата процесса (параметр ExitProcess )
call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры

```

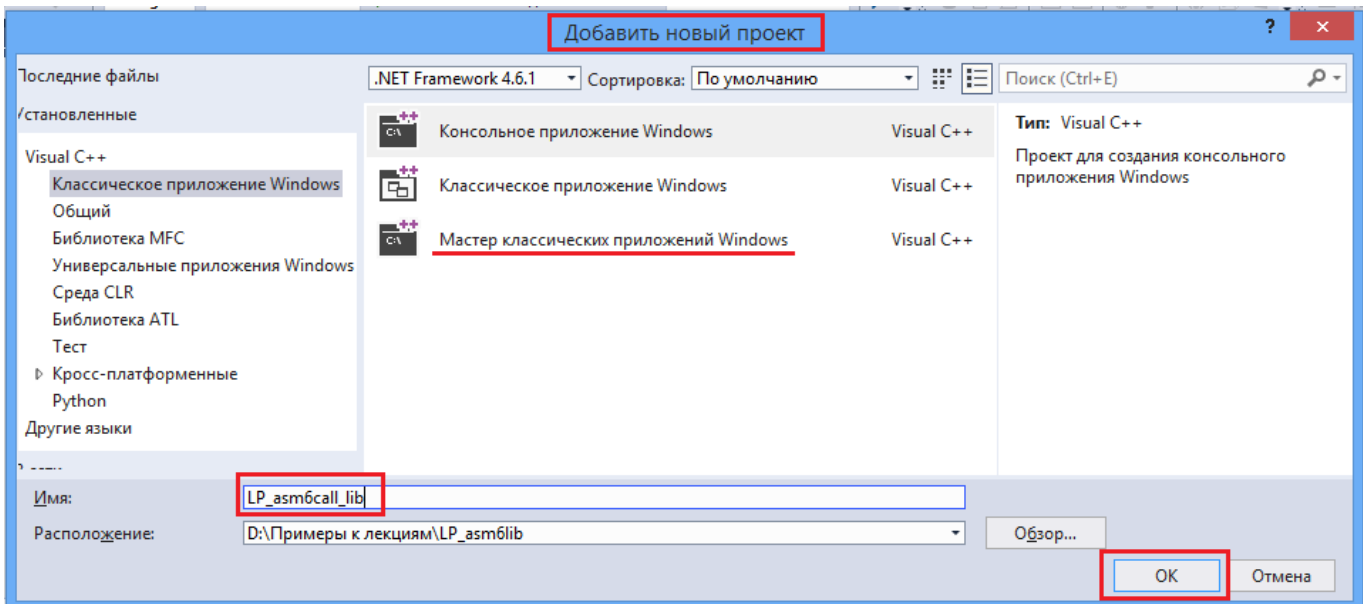
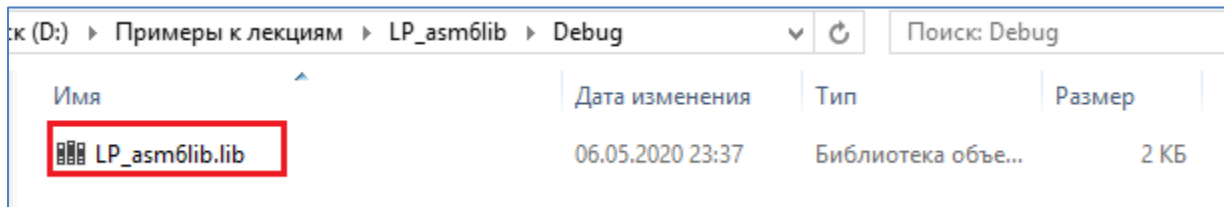
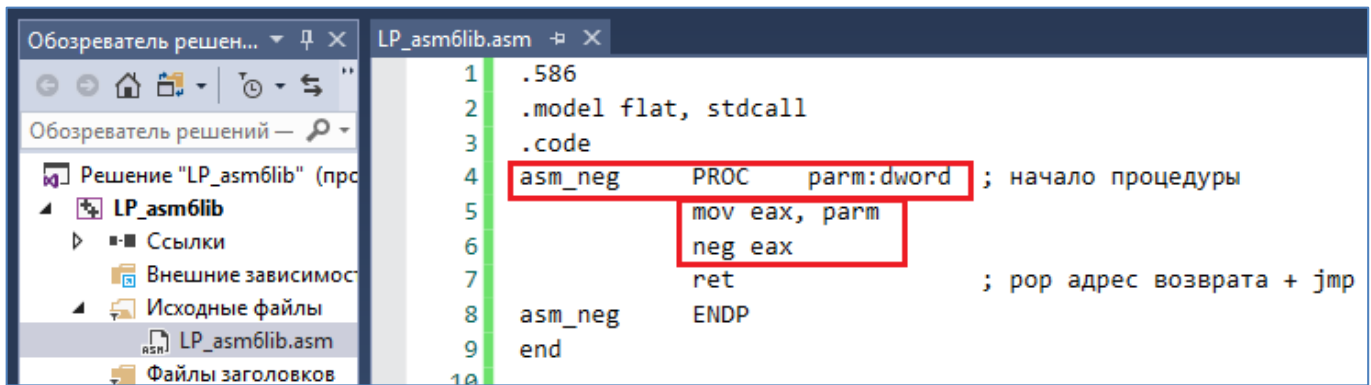


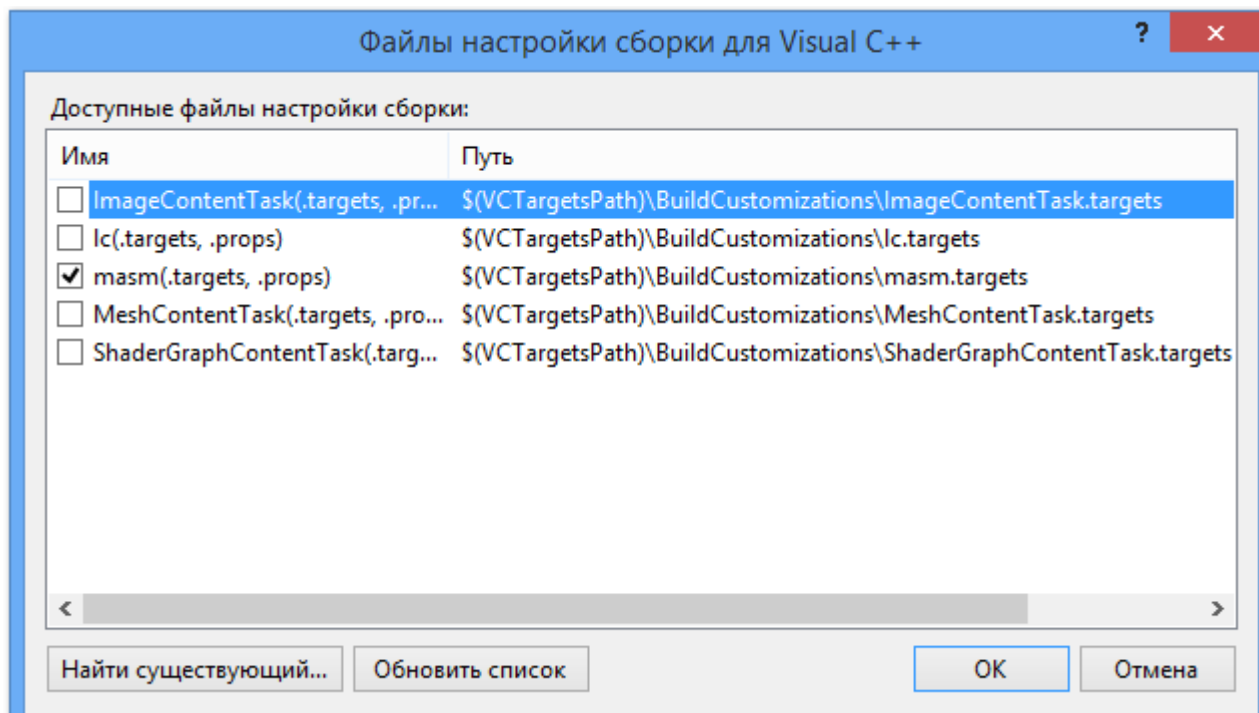
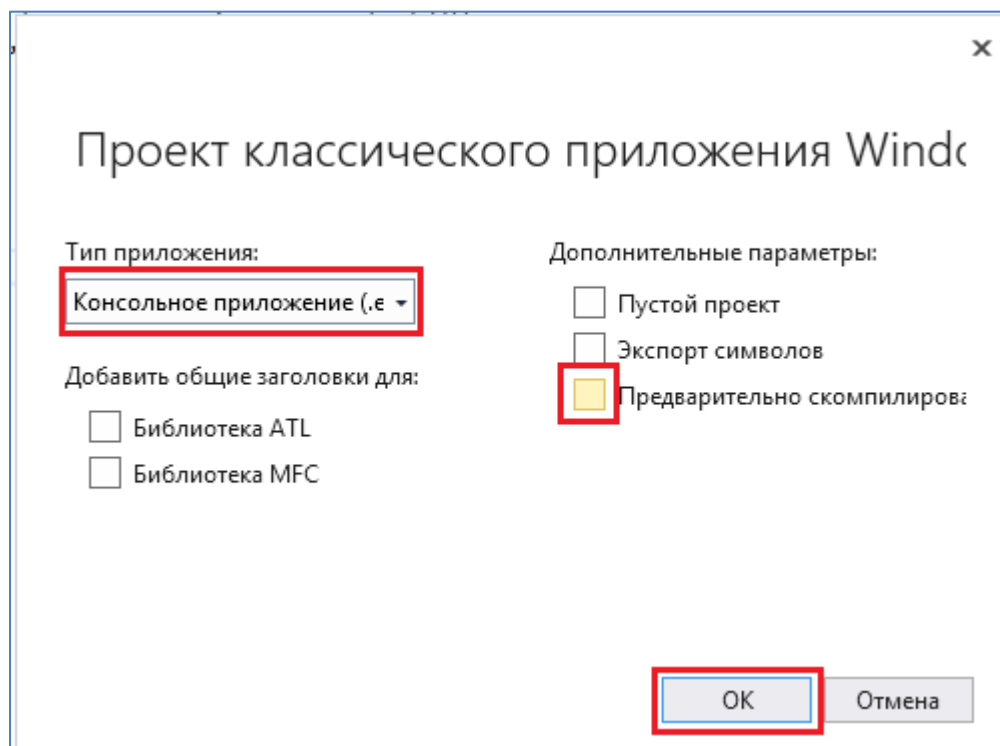
### 3. Создание статической библиотеки

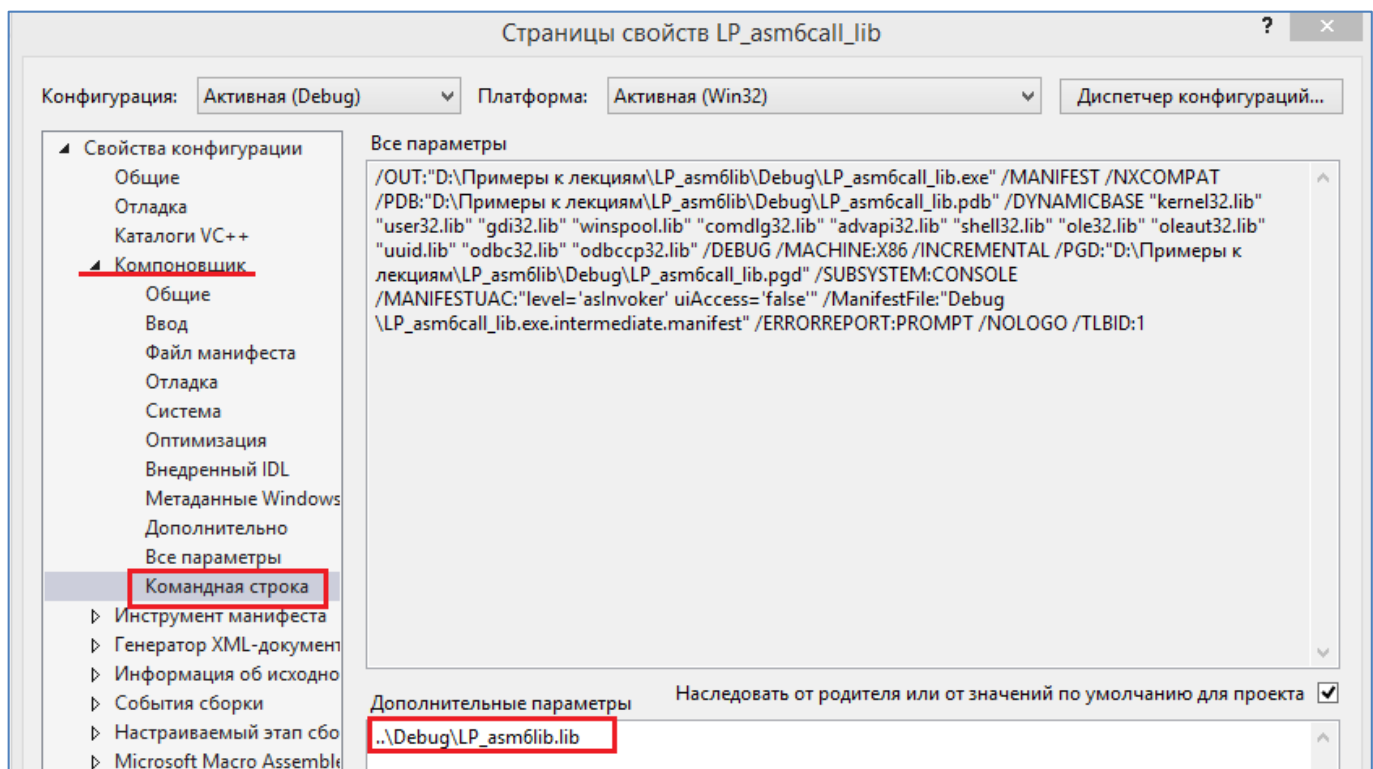
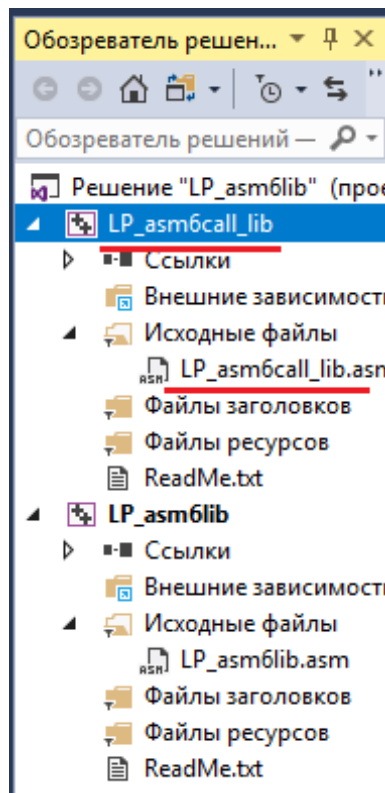












## а. Вызов функции из C++

Решение "LP\_asm6lib" (проектов: 3)

LP\_asm6call\_lib

- Ссылки
- Внешние зависимости
- Исходные файлы
  - LP\_asm6call\_lib.asm
- Файлы заголовков
- Файлы ресурсов
- ReadMe.txt

LP\_asm6lib

- Ссылки
- Внешние зависимости
- Исходные файлы
  - LP\_asm6lib.asm
- Файлы заголовков
- Файлы ресурсов
- ReadMe.txt

```
1 .586
2 .model flat, stdcall
3 includelib kernel32.lib
4 ExitProcess PROTO :DWORD
5
6 includelib LP_asm6lib.lib
7 asm_neg PROTO :DWORD
8
9 .stack 4096
10 .const
11 .data
12 .code
13 main PROC ; начало процедуры
14
15 push -777
16 call asm_neg
17
18 push 0
19 call ExitProcess
20 main ENDP
21 end main
```

Контрольные значения 1

Имя	Значение
eax	777

## б. Вызов функции из C++

Решение "LP\_asm6lib" (проектов: 3)

- LP\_asm6call\_lib
- LP\_asm6lib
- LP\_cpp6call\_lib

LP\_cpp6call\_lib

- Ссылки
- Внешние зависимости
- Исходные файлы
  - LPcpp6calllib.cpp
- stdafx.cpp
- Файлы заголовков
  - stdafx.h
  - targetver.h
- Файлы ресурсов
- ReadMe.txt

```
1 // LPcpp6calllib.cpp: определяет точку входа для консольного
2 //
3
4 #include "stdafx.h"
5
6 extern "C"
7 {
8     int __stdcall asm_neg(int);
9 }
10
11 int main()
12 {
13     int x = asm_neg(7777);
14     return 0;
15 }
16
```

Контрольные значения 1

Имя	Значение	Тип
x	-7777	int

По умолчанию интерфейсы согласуются по правилам C++.

Расширение имени (декорирование) позволяет компоновщику различить перегружаемые функции с одинаковыми именами, но разными параметрами (декорирование имен отсутствует в языке C).

Модификатор Extern "C" запрещает декорирование имен.