

БГТУ, ФИТ, ПОИТ, 2 семестр, Языки программирования

Введение в язык Ассемблер

1. Вывод в консоль

1.1 Windows API-функция **SetConsoleTitleA** – устанавливает заголовок (тайтл) для текущего консольного окна.

Синтаксис:

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

Требуемая библиотека: Kernel32.lib

Соглашение о вызовах: stdcall

Входной параметр: 32-разрядный указатель на строку, которую будет выведена в заголовке окна. Строка должна заканчиваться нулем. Общий размер должен быть меньше 64К.

Возвращаемое значение: { 0 – функция завершается с ошибкой;
иначе – функция завершается успешно.

Буква в конце функции означает:

"A" – вывод строки в формате ASCII (один символ – один байт);

"W" (W – от wide) – в Unicode (один символ – два байта).

Пример:

```

includelib kernel32.lib      ; компоновщику: компоновать с kernel32.lib
ExitProcess      PROTO      :DWORD ; прототип функции

SetConsoleTitleA PROTO      :DWORD ; установить заголовок окна консоли (ANSI)

.stack 4096                ; сегмент стека объемом 4096
.const                    ; сегмент констант

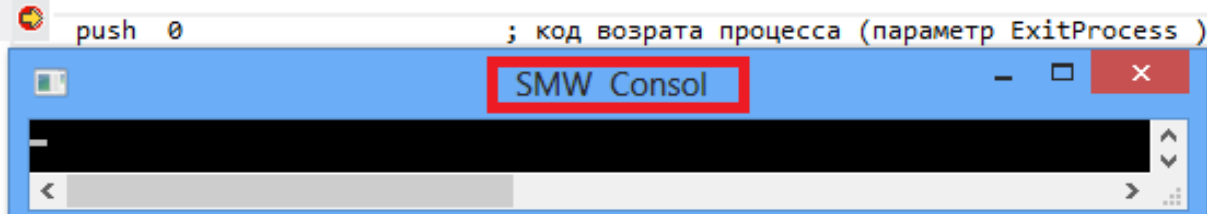
consoletitle db ' SMW  Consol',0

.data                      ; сегмент данных
.code                      ; сегмент кода
main PROC                  ; начало процедуры

push offset consoletitle   ; параметр SetConsoleTitleA: адрес строки
call SetConsoleTitleA      ; вызов SetConsoleTitleA

push 0                     ; код возврата процесса (параметр ExitProcess )


```



The screenshot shows a Windows console window with a blue title bar. The title bar text is "SMW Consol", which is highlighted with a red rectangle. The console area is black with white text, showing a single line of text that is mostly obscured by the title bar. The window has standard Windows controls (minimize, maximize, close) on the right side.

- 1.2 Функция **GetStdHandle** извлекает дескриптор для устройства стандартного ввода данных, стандартного вывода данных или стандартной ошибки.

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

- 1.3 Функция **WriteConsole** выводит символьную строку в консоль, начиная с текущей позиции курсора.

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 |
|); | |

Пример:

```
ExitProcess      PROTO :DWORD ; прототип функции
SetConsoleTitleA PROTO :DWORD ; установить заголовок окна консоли (ANSI)
GetStdHandle     PROTO :DWORD ; получить handle вывода на консоль
WriteConsoleA    PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD ; вывод на консоль
```

```
.stack 4096          ; сегмент стека объемом 4096
.const              ; сегмент констант
```

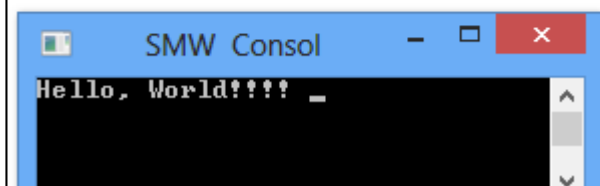
```
consoletitle db 'SMW Consol',0
helloworld   db 'Hello, World!!!!',0
```

```
.data                ; сегмент данных
consolehandle dd 0h   ; handle консоли
.code                ; сегмент кода
main PROC            ; начало процедуры
```

```
push offset consoletitle ; параметр SetConsoleTitleA: адрес строки
call SetConsoleTitleA    ; вызов SetConsoleTitleA
```

```
push -11              ; -11 - handle для стандартного вывода
call GetStdHandle      ; получить handle->eax
mov consolehandle, eax
```

```
push 0                 ; можно 0
push 0                 ; можно 0
push sizeof helloworld ; количество выводимых байт
push offset helloworld ; адрес выводимой строки
push consolehandle     ; handle для вывода
call WriteConsoleA     ; вывести на консоль
```



рата процесса (параметр ExitProcess)
жен заканчиваться любой процесс Windows
процедуры
одуля, main - точка входа

2. Процедура вывода

Вызов процедуры:

```
.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 'SMW  Consol',0
str_helloworld  db 'Hello, World!!!!',10,0
str_pause       db 'pause',0
.data                                                  ; сегмент данных

.code                                                  ; сегмент кода
main PROC                                             ; начало процедуры

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

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

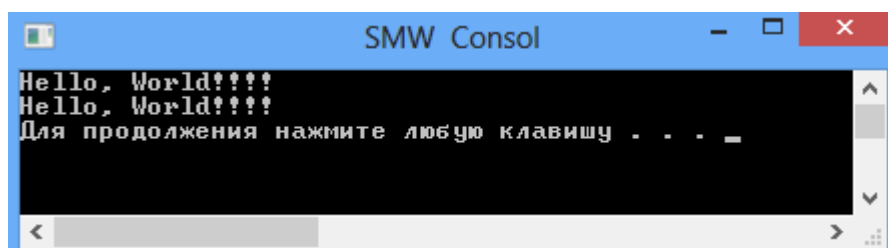
push offset str_pause      ; адрес выводимой cmd-команды
call system                ; system("pause");

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

main ENDP                 ; конец процедуры
```

Процедура вывода сообщения в окно консоли:

```
; -----  
;  
;                               Вывод сообщения в консоль  
; -----  
    includelib kernel32.lib  
    SetConsoleTitleA    PROTO    :DWORD    ; установить заголовок окна консоли (ANSI)  
    GetStdHandle        PROTO    :DWORD    ; получить handle вывода на консоль  
    WriteConsoleA       PROTO    :DWORD, :DWORD, :DWORD, :DWORD, :DWORD ; вывод на консоль  
  
    printconsole PROC uses eax ebx ecx edi esi,  
                    pstr: dword, ptitle: dword  
  
        push ptitle          ; параметр SetConsoleTitleA: адрес строки  
        call SetConsoleTitleA ; вызов SetConsoleTitleA  
  
        push -11             ; -11 - handle для стандартного вывода  
        call GetStdHandle     ; получить handle->eax  
  
        mov esi, pstr        ; подсчет количества  
        mov edi, -1          ; символов (до 0h)  
count:                                     ; в выводимой  
        inc edi              ; на консоль строке  
        cmp byte ptr [esi + edi], 0 ;  
        jne count            ; количество символов ->edi  
  
        push 0               ; можно 0  
        push 0               ; можно 0  
        push edi  
        push pstr            ; адрес выводимой строки  
        push eax             ; handle для вывода  
        call WriteConsoleA   ; вывести на консоль  
  
        ret  
    printconsole ENDP  
  
    end main                ; конец модуля, main - точка входа
```



3. Процедура преобразования числа в символы

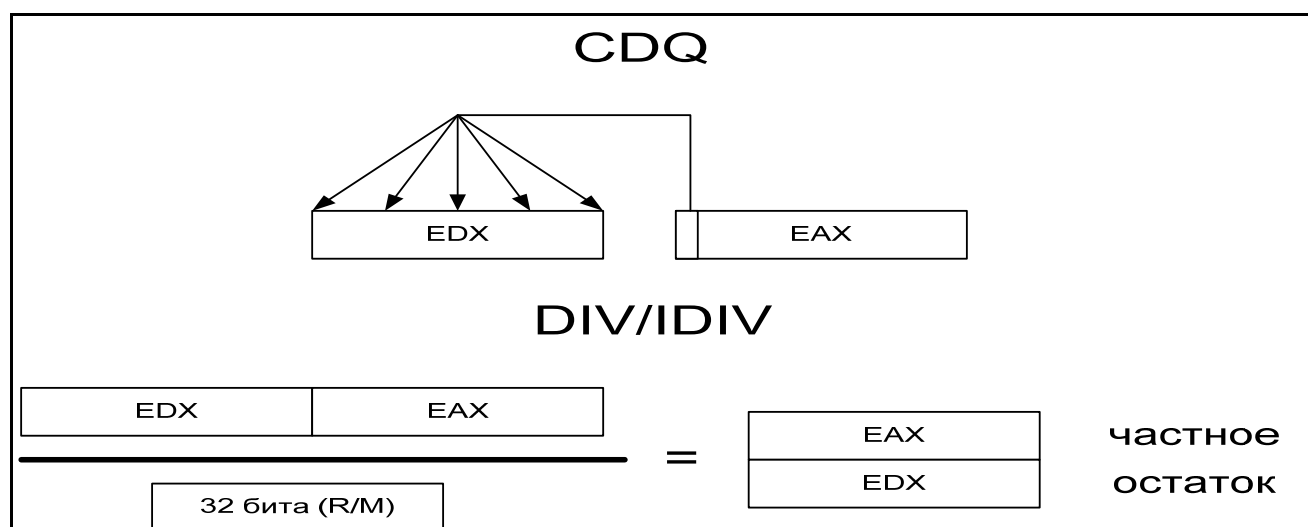
```

; -----
;                               Преобразование числа в строку
; -----
int_to_char  PROC uses eax ebx ecx edi esi,
                pstr: dword,          ; адрес строки-результата
                intfield: dword       ; преобразуемое число

    mov edi, pstr                ; адрес результата -> edi
    mov esi, 0                   ; количество символов в результате
    mov eax, intfield            ; число -> eax
    cdq                         ; знак распространили на с eax на edx
    mov ebx, 10                  ; десятичная система счисления
    idiv ebx                     ; аех = eax/ebx, остаток->edx
    test eax, 80000000h          ; результат отрицательный ?
    jz plus                      ; если положительный на plus
    neg eax                      ; eax = - eax
    neg edx                      ; edx = -edx
    mov cl, '-'                  ; первый символ результата '-'
    mov [edi], cl                ; первый символ результата '-'
    inc edi                      ; ++edi
plus:                          ; цикл разложения на степени 10
    push dx                      ; остаток -> стек
    inc esi                      ; ++esi
    test eax, eax                ; eax == 0?
    jz fin                      ; если да то на fin
    cdq                          ; знак распространили на с eax на edx
    idiv ebx                     ; аех = eax/ebx, остаток->edx
    jmp plus                    ; переход на plus
fin:
    mov ecx, esi                 ; количество не 0вых остатков = количеству символов в результате
write:                          ; цикл записи результата
    pop bx                      ; остаток из стека -> bx
    add bl, '0'                  ; сформировали символ в bl
    mov [edi], bl                ; bl-> в результат
    inc edi                      ; edi++
    loop write                   ; if (--ecx) > 0 goto write

    ret
int_to_char ENDP
; -----

```



Пример:

```
.586 ; система команд (процессор Pentium)
.model flat, stdcall ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компоновщику: 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 -77777777 ; исходное число
push offset result1 ; место для результата
call int_to_char ; вызов процедуры преобразования

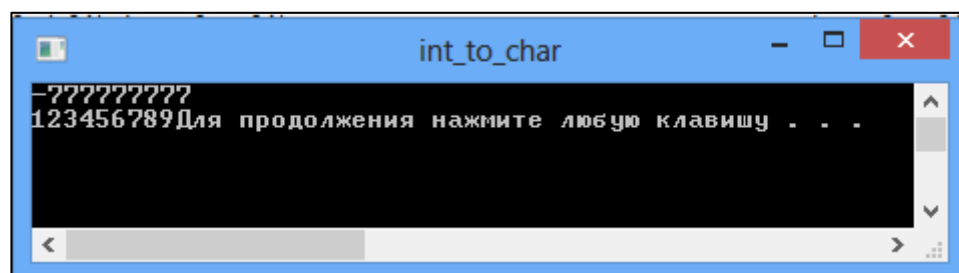
push offset consoletitle ; заголовок окна консоли
push offset result1 ; выводимый текст
call printconsole ; вызов процедуры

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
```



4. Пример консольного приложения на ассемблере (смотри демо на diskstation)

```
.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
;-----
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 конст. значение - 10 ввод,
; -11 вывод,
; -12 ошибка устройства вывода)
WriteConsoleA PROTO : DWORD, : DWORD, : DWORD, : DWORD, : DWORD
; вывод на консоль(стандартная функция)
;-----
printconsole PROTO : DWORD, : DWORD ; вызов процедуры 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
.const ; сегмент констант
endl equ 0ah ; символ перевода строки (ASCII)
str_endl byte endl,0 ; строка "конец строки"
;-----
.data
result1 byte "Результат вызова процедуры int_to_char = "
result byte 40 dup(0)
; строка заголовка, первый элемент данные + нулевой байт
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 ?
begin byte "Тема: Определение и использование процедур в MASM",0
str_pause byte "pause", 0 ;
consolehandle dword 0h ; состояние консоли

.code
; -----MAIN-----
main PROC ; точка входа main
push offset consoleTitle ; в стек параметр: смещение строки
call SetConsoleTitleA ; вызов функции установки заголовка окна
;-----
```



```

        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
;-----
        invoke printconsole, offset str_endl, offset consoleTitle
        ; вывод конца строки
;-----
        push 1251d
        call SetConsoleOutputCP
        push 1251d
        call SetConsoleCP
        invoke printconsole, offset begin, offset consoleTitle ; тема лекции
;-----
        invoke printconsole, offset str_endl, offset consoleTitle ; конец строки
;-----
        invoke printconsole, offset str_endl, offset consoleTitle ; конец строки
;-----
        push -777                ; исходное число
        push offset result        ; где результат
        call int_to_char          ; вызов процедуры преобразования
;-----
        invoke printconsole, offset result1, offset consoleTitle ; результат
;-----
        invoke printconsole, offset str_endl, offset consoleTitle ; конец строки
        invoke printconsole, offset str_endl, offset consoleTitle ; конец строки
;-----
        push offset str_pause    ; адрес выводимой cmd команды
        call system              ; system("pause")
;-----
        push 0
        call ExitProcess         ; завершение процесса Windows
main    ENDP                    ; конец процедуры main

; -----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
        mov esi, pstr            ; подсчет количества символов
        mov edi, -1              ; до 0-символа

```

```

count:                                ; выводимой
    inc edi                            ; на консоль строки
    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

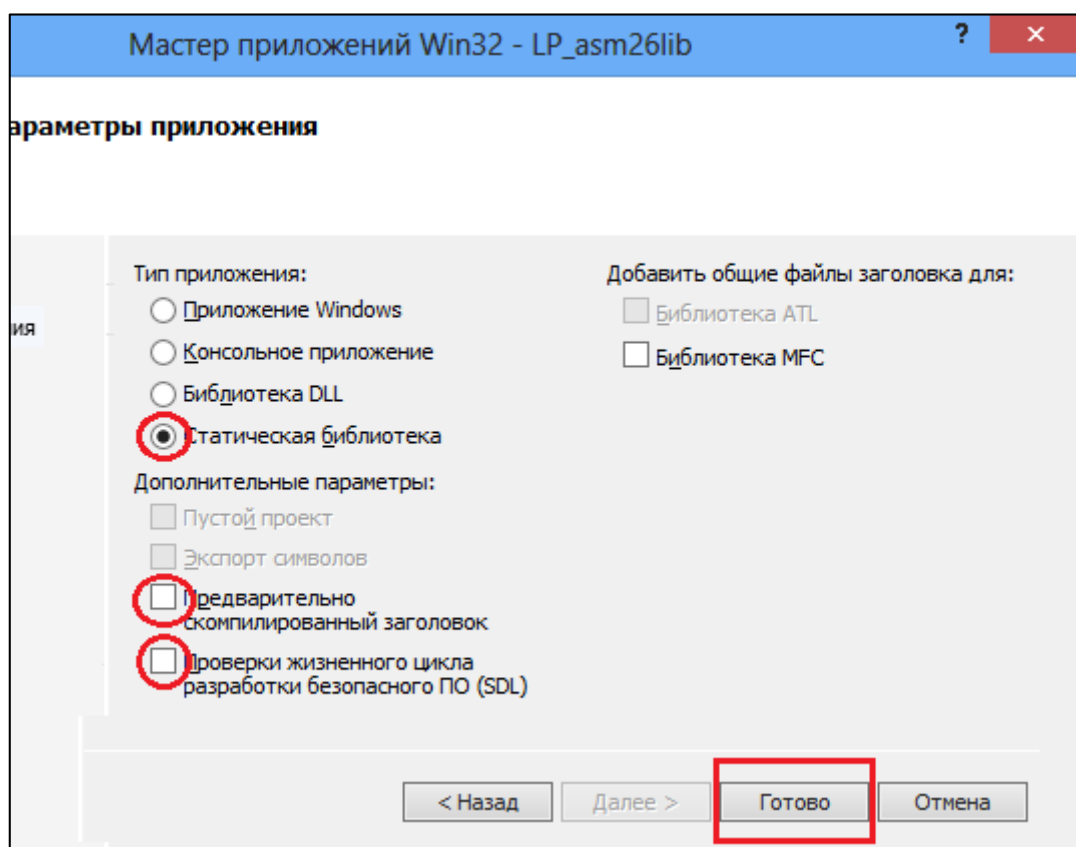
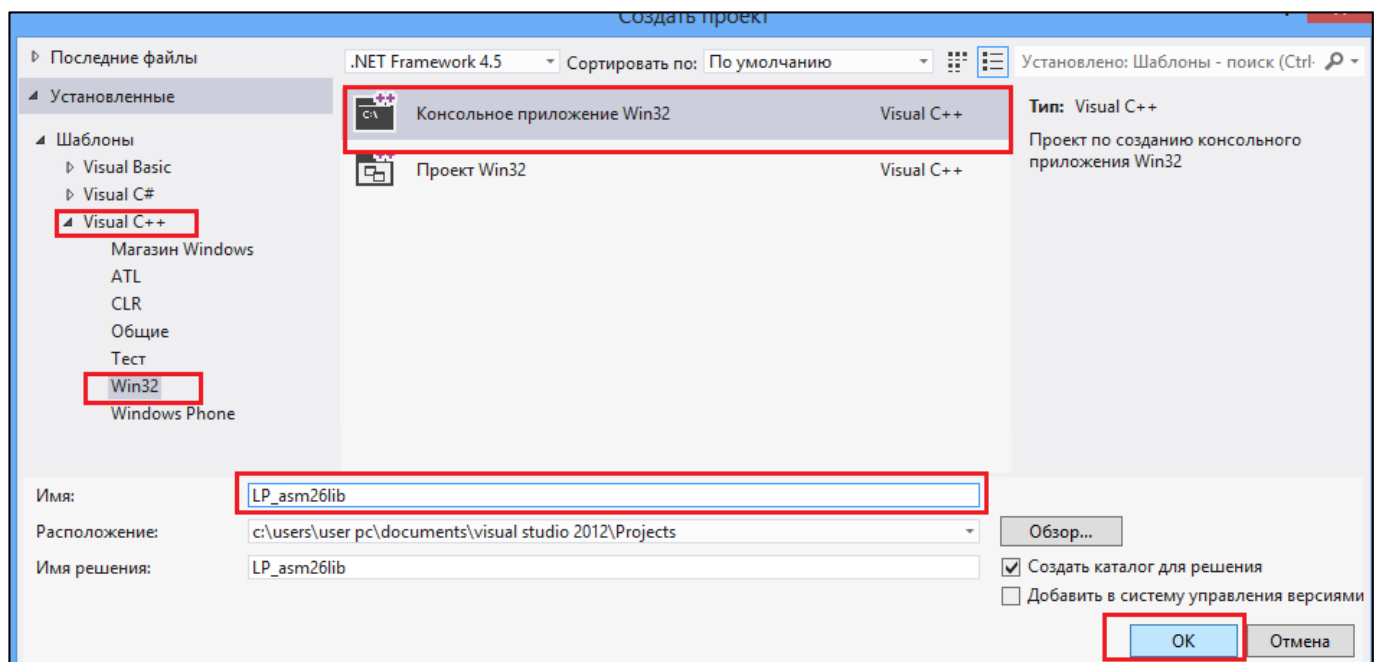
printconsole    ENDP

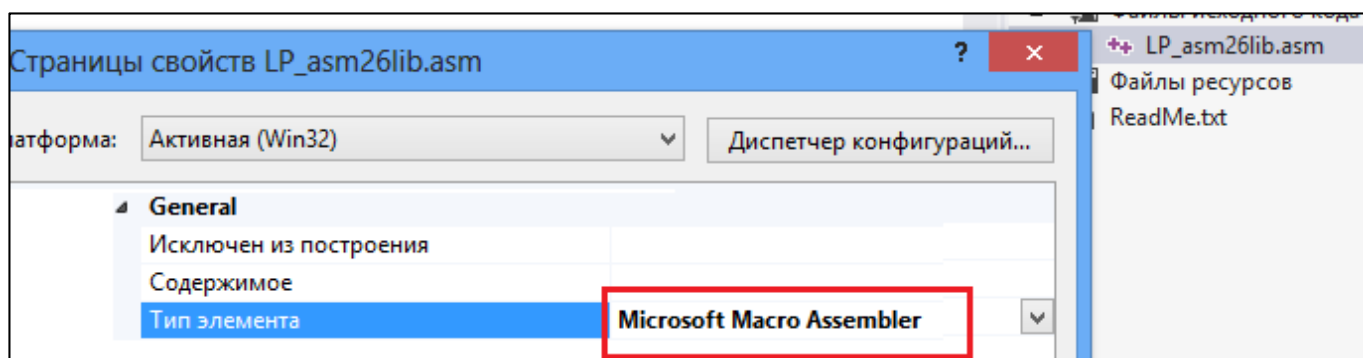
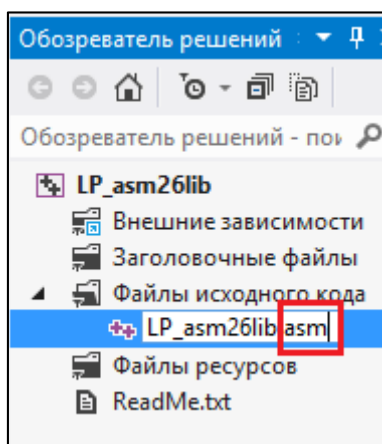
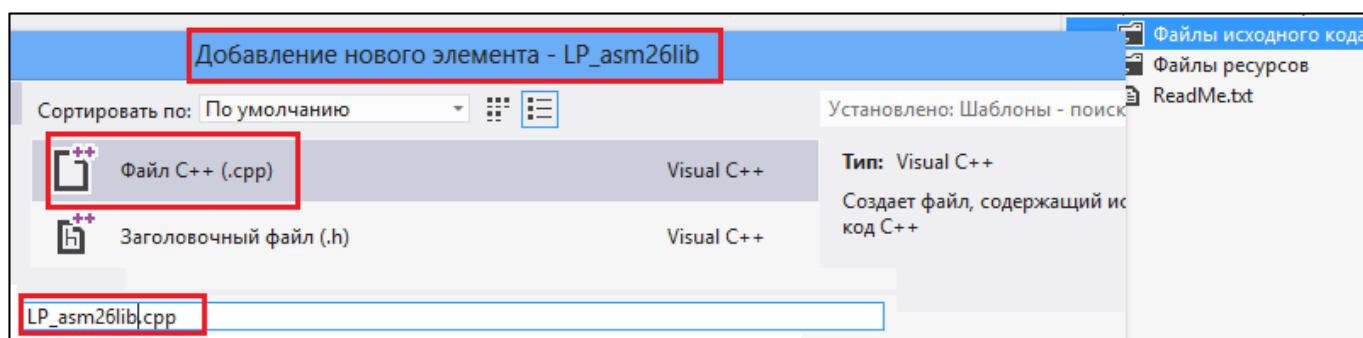
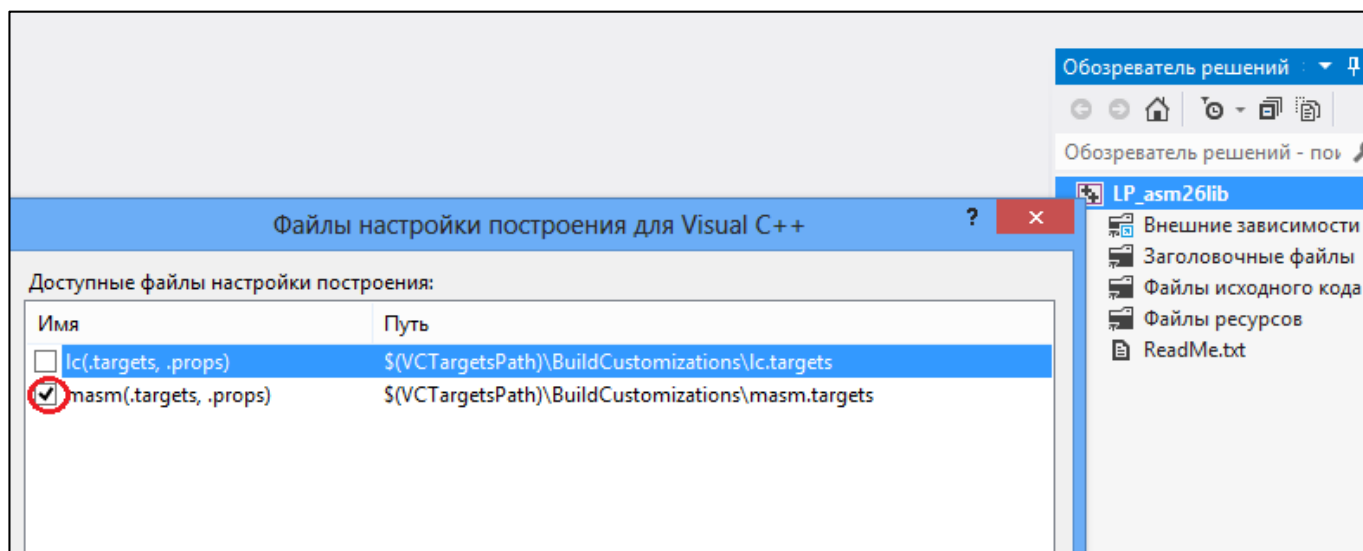
; -----преобразование числа в строку-----
int_to_char     PROC uses eax ebx ecx edi esi,
                pstr : dword,          ; адрес строки результата
                intfield : sdword      ; число для преобразования

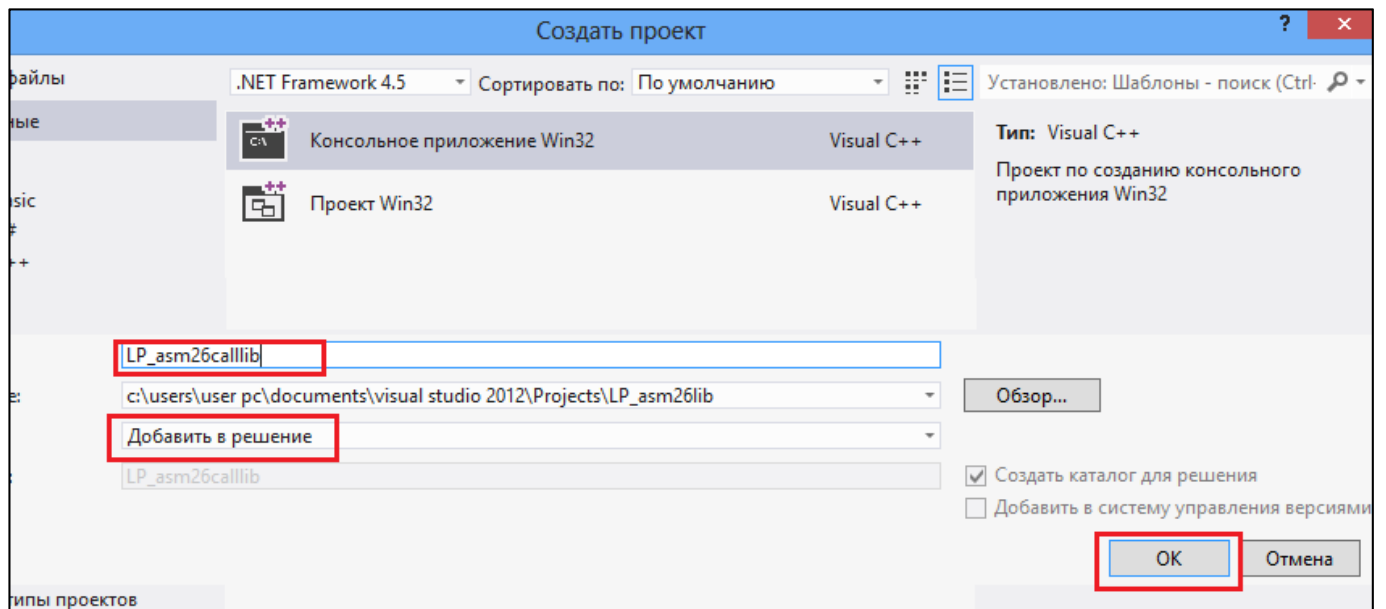
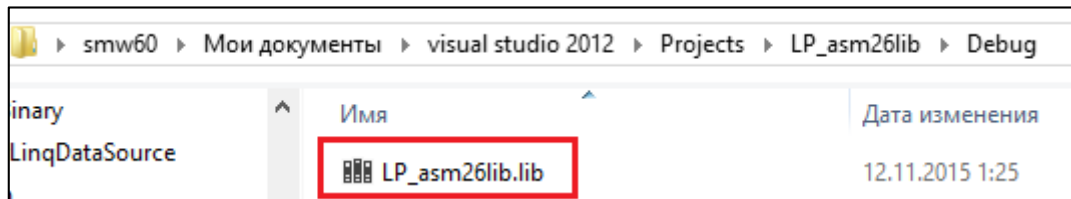
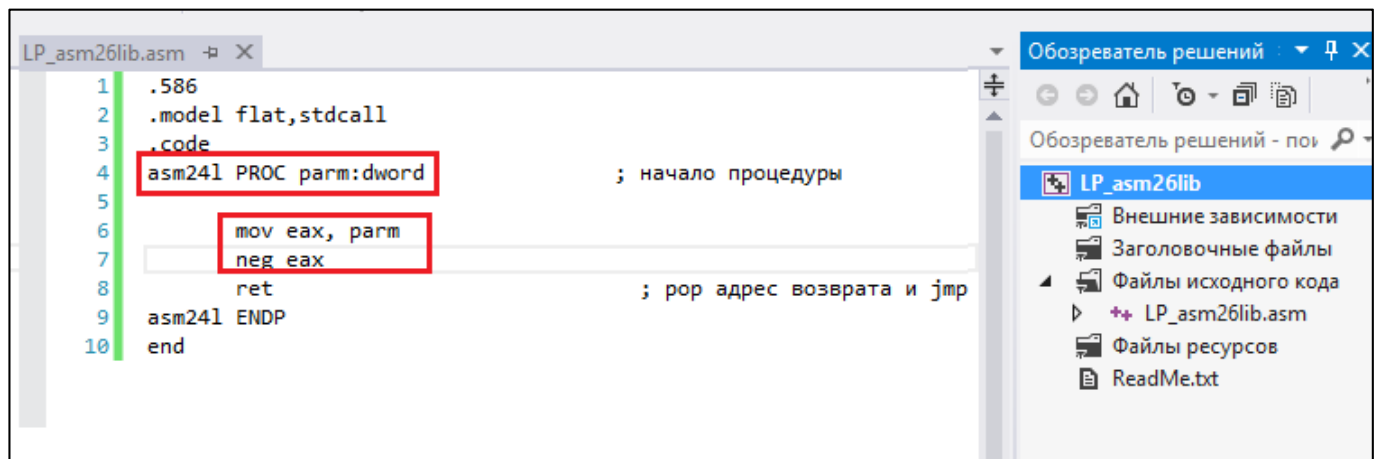
    mov edi, pstr                      ; копирует из pstr в edi
    mov esi, 0                        ; количество символов в результате
    mov eax, intfield                 ; число -> в eax
    cdq                              ; знак числа распространяется с eax на edx
    mov ebx, 10                       ; 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 :
    ; в ecx кол-во не 0-вых остатков = кол-ву символов результата
    mov ecx, esi
write :
    ; цикл записи результата
    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
end             main                  ; конец модуля main

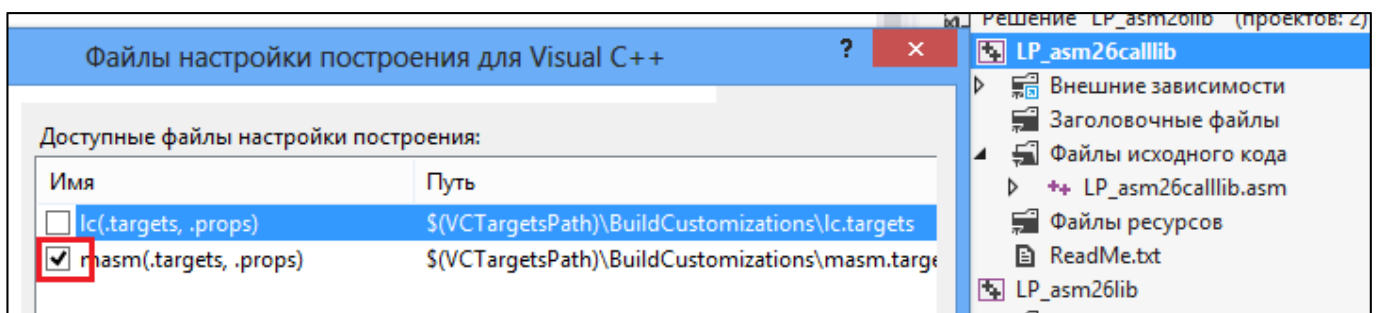
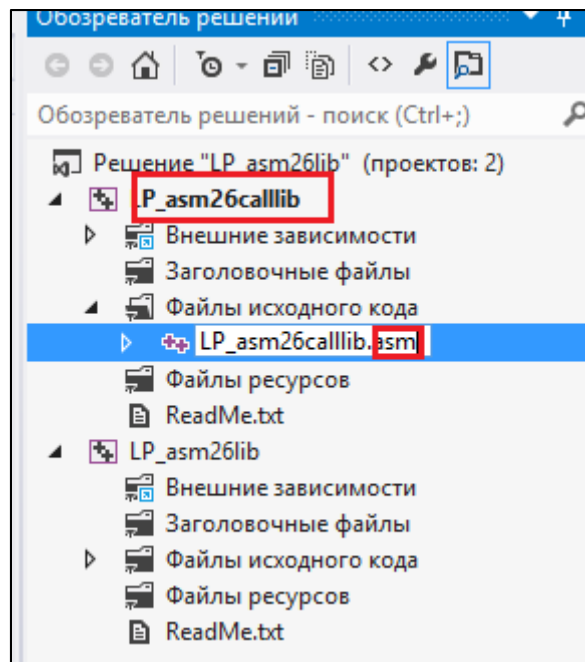
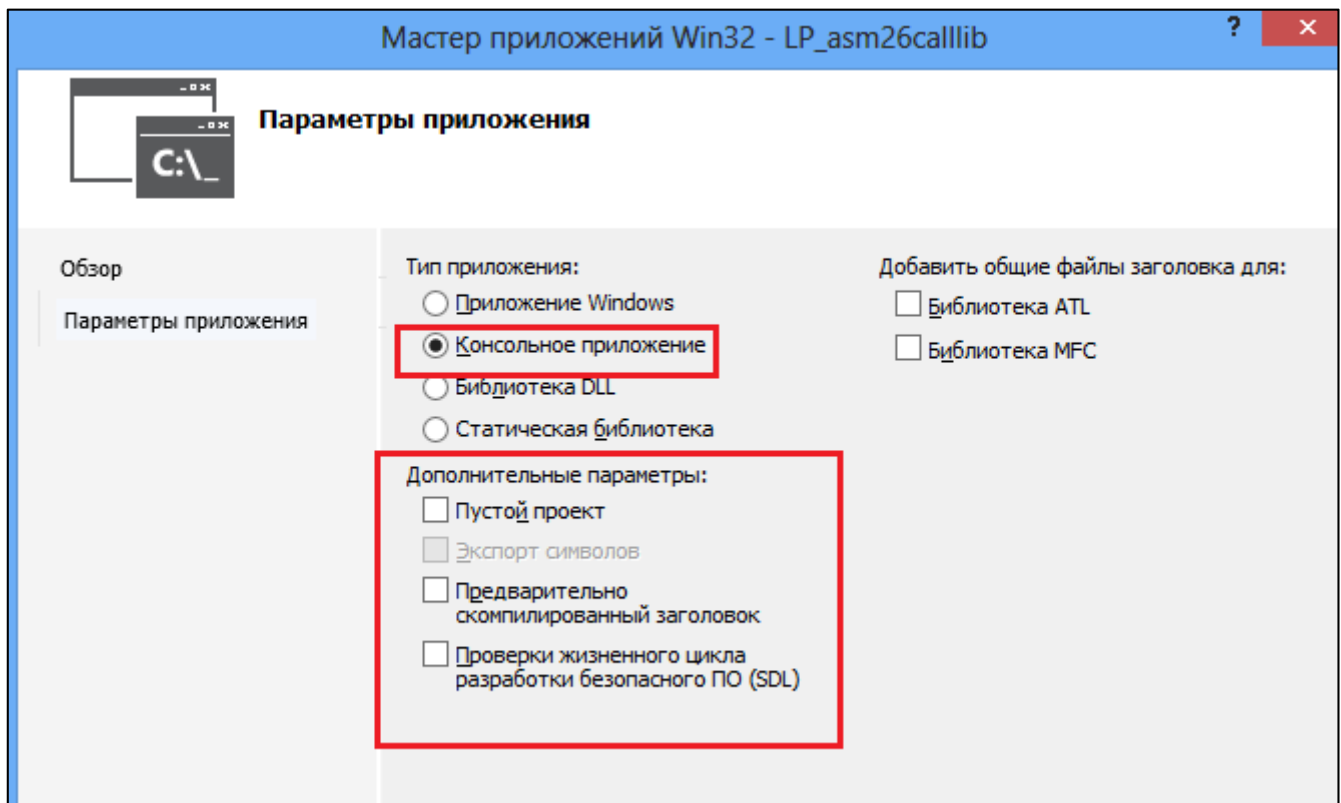
```

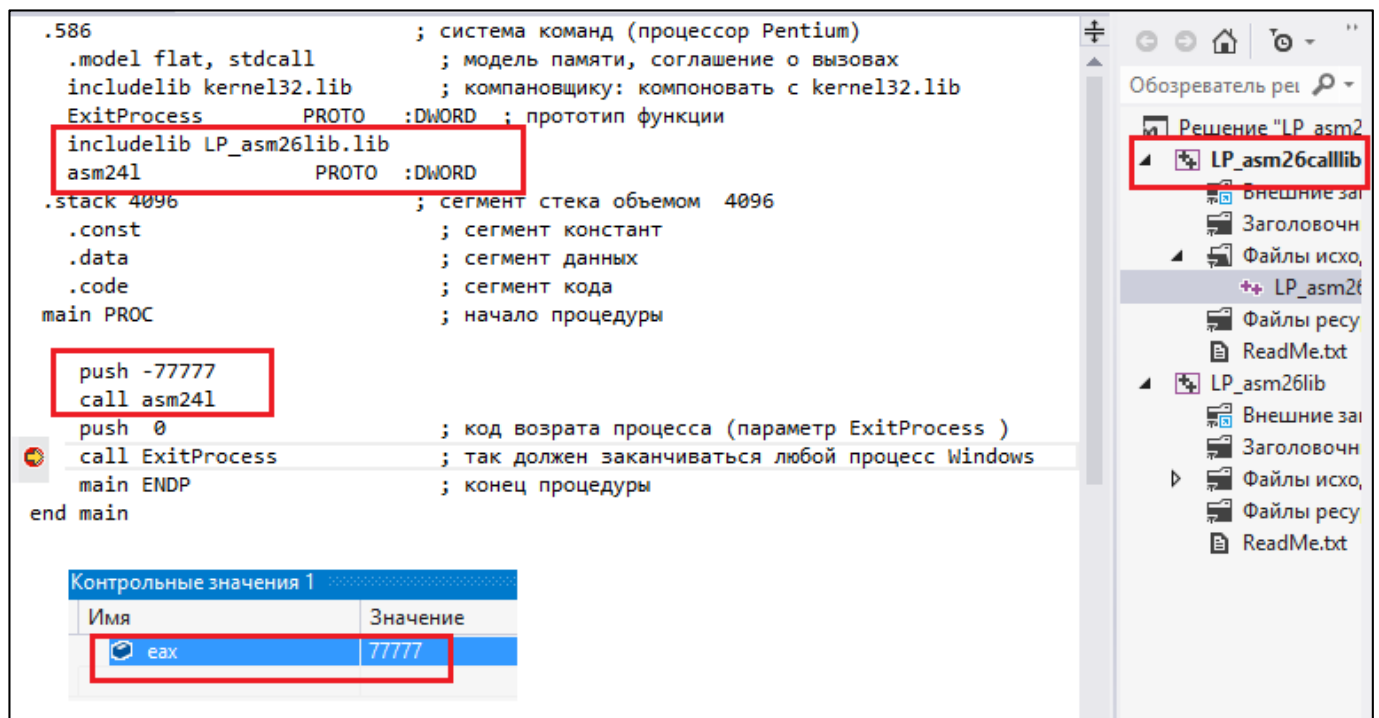
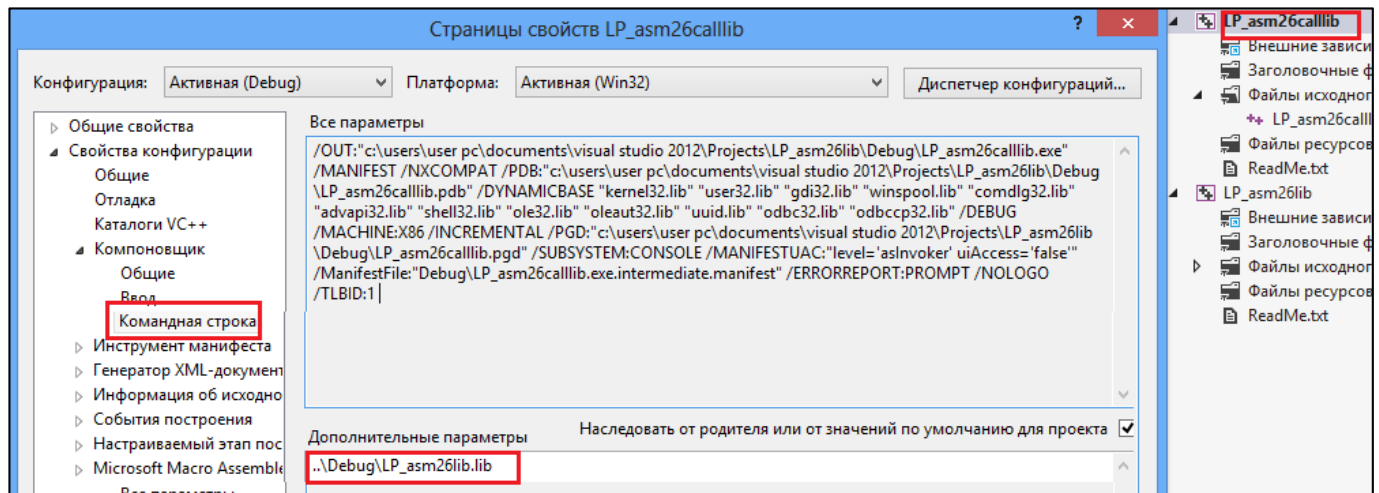
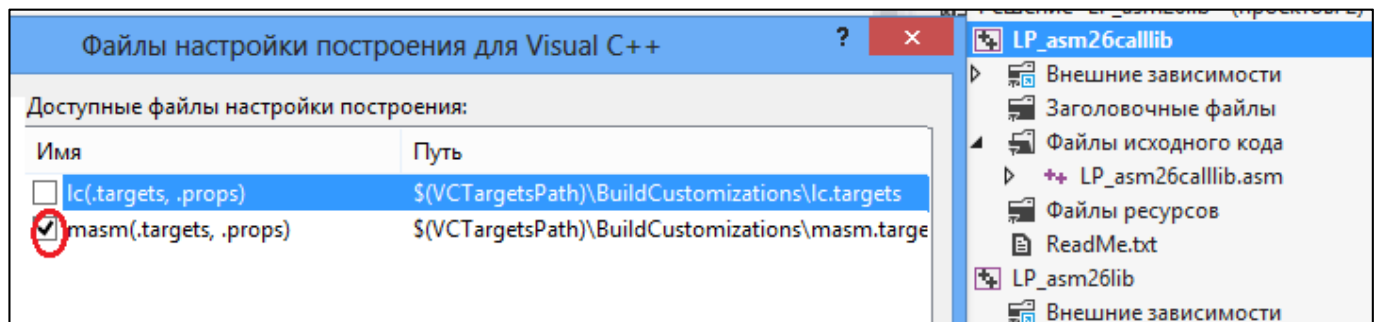
5. Создание статической библиотеки











6. Вызов функции из C++

The screenshot displays a C++ development environment with three main components:

Code Editor: The code in `LP_cpp26callib.cpp` is as follows:

```
1 // LP_cpp26callib.cpp: определяет точку входа для приложения
2 //
3
4 #include "stdafx.h"
5
6 extern "C"
7 {
8     int __stdcall asm241(int x);
9 };
10
11 int _tmain(int argc, _TCHAR* argv[])
12 {
13     int x = asm241(7777);
14
15     return 0;
16 }
17
18
19
```

Solution Explorer: The project structure is shown on the right:

- Решение "LP_asm26lib" (проектов: 3)
 - LP_asm26calllib
 - LP_asm26lib
 - LP_cpp26callib**
 - Внешние зависимости
 - Заголовочные файлы
 - stdafx.h
 - targetver.h
 - Файлы исходного кода
 - LP_cpp26callib.cpp**
 - stdafx.cpp
 - Файлы ресурсов
 - ReadMe.txt

Контрольные значения 1: A variable watch window at the bottom shows the value of `x`:

| Имя | Значение |
|-----|----------|
| x | -7777 |