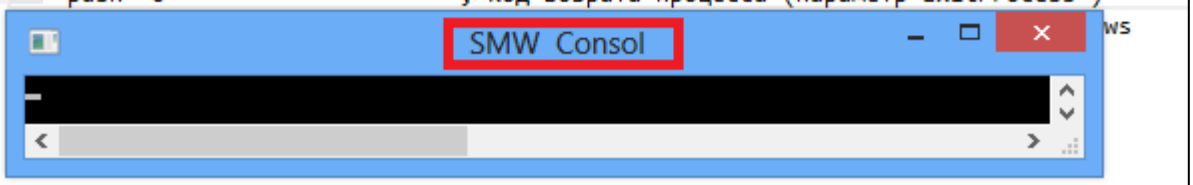
Лекция А6

БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования

Введение в язык Ассемблер

1. Вывод в консоль

```
.586                            ; система команд (процессор Pentium)
.model flat,stdcall             ; модель памяти, соглашение о вызовах
includelib kernel32.lib         ; компановщику: 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 возрата процесса (параметр ExitProcess )
```



Синтаксис:

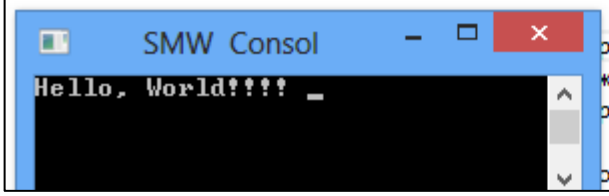
```
BOOL SetConsoleTitle(       // устанавливает заголовок для консольного окна
  LPCTSTR lpConsoleTitle    // 32-разрядный  указатель на строку lpConsoleTitle –
                            // заголовок консольного окна
);
```

Необходимая библиотека: Kernel32.lib

Соглашение о вызовах: stdcall

Возвращаемые значения: $\begin{cases} 0 - \text{функция завершается с ошибкой;} \\ \text{иначе} - \text{функция завершается успешно.} \end{cases}$

```
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
роцедуры

одуля, main - точка входа
```

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

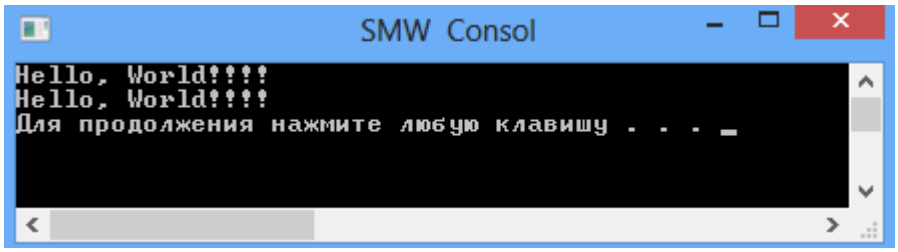
Синтаксис:

```
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**

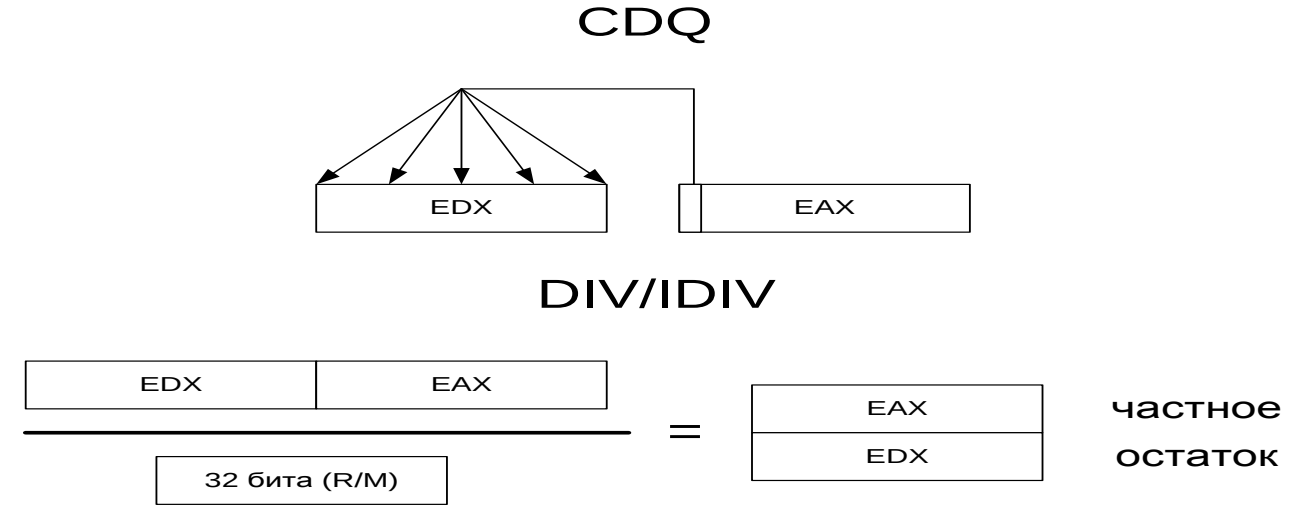
```
.586                            ; система команд (процессор Pentium)
.model flat,stdcall             ; модель памяти, соглашение о вызовах

includelib kernel32.lib         ; компановщику: 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 возрата процесса (параметр ExitProcess )
 call ExitProcess               ; так должен заканчиваться любой процесс Windows

 main ENDP                      ; конец процедуры
```

```
; ------------------------------------------------------------------------------
;                          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                     ;  количетво 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
```

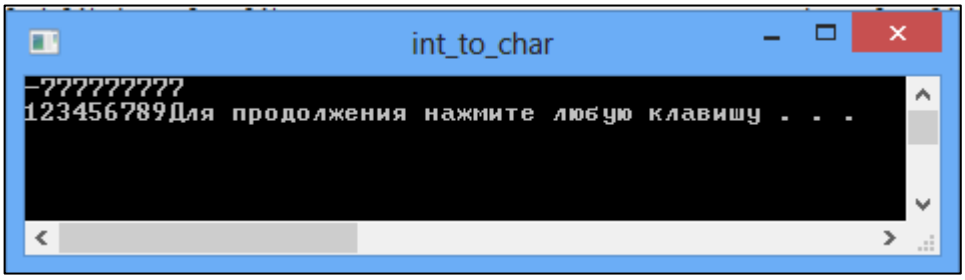


3. **Процедура преобразования числа в символы**

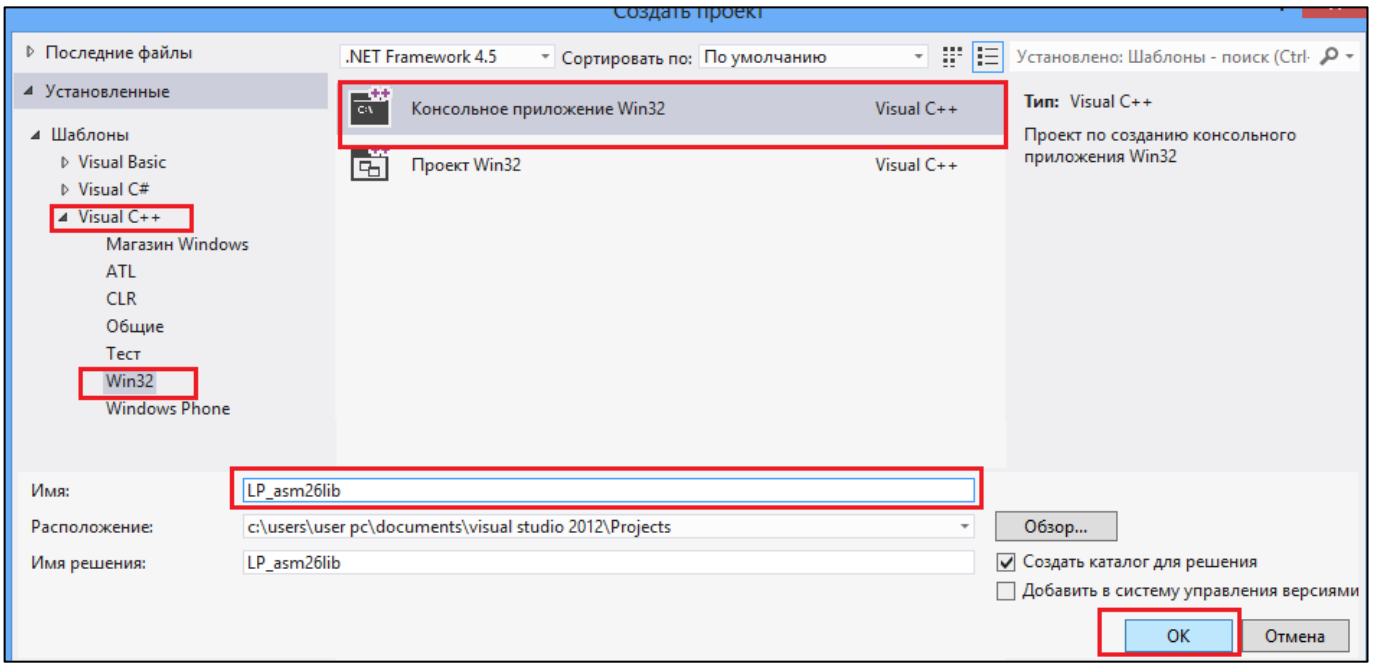
```
; ------------------------------------------------------------------------
;                          Преобразование числа в строку
;------------------------------------------------------------------------
 int_to_char   PROC uses eax ebx ecx edi  esi,
                        pstr: dword,              ; 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                     ; aex = 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                     ; aex = 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
 ; ------------------------------------------------------------------------
```

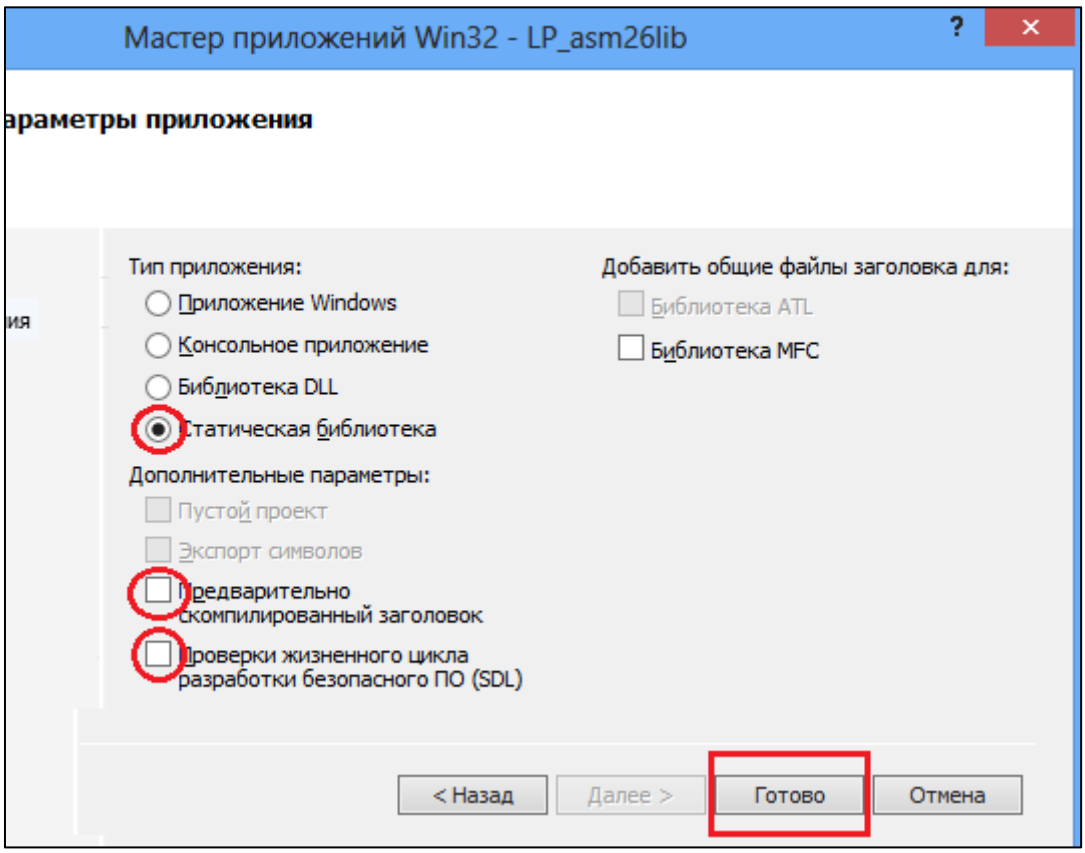


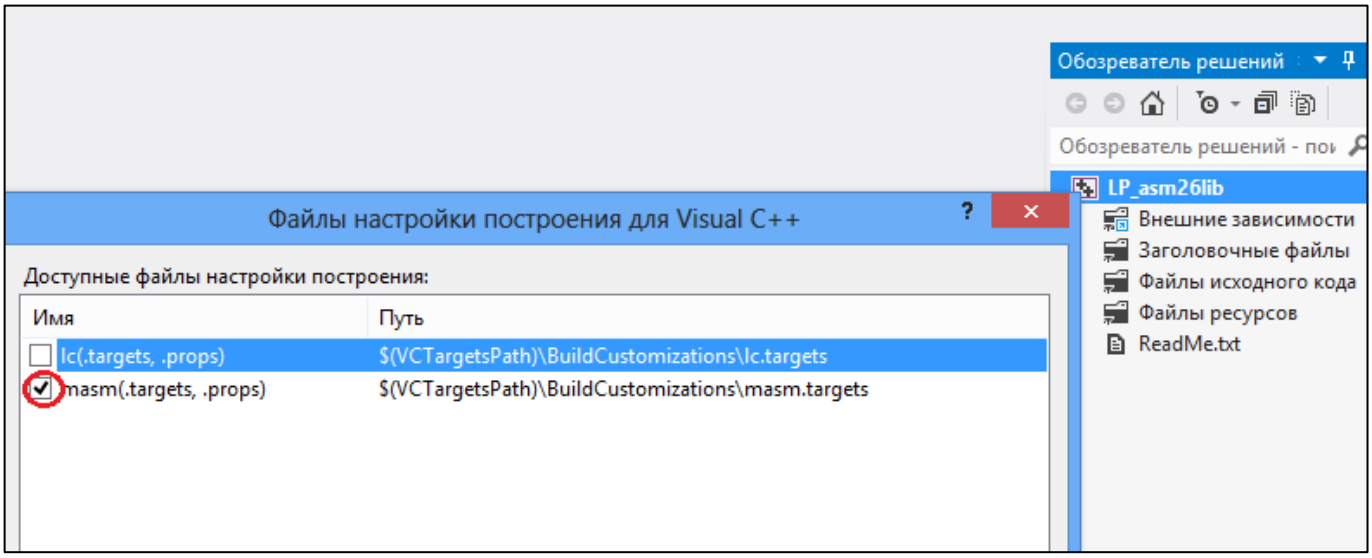
```
  .586                          ; система команд (процессор Pentium)
  .model flat, stdcall          ; модель памяти, соглашение о вызовах
  includelib kernel32.lib       ; компановщику: компоновать с kernel32.lib
  ExitProcess        PROTO   :DWORD  ; прототип функции
  includelib msvcrt.lib         ; 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 возрата процесса (параметр ExitProcess )
  call ExitProcess              ; так должен заканчиваться любой процесс Windows
  main ENDP                     ; конец процедуры
```

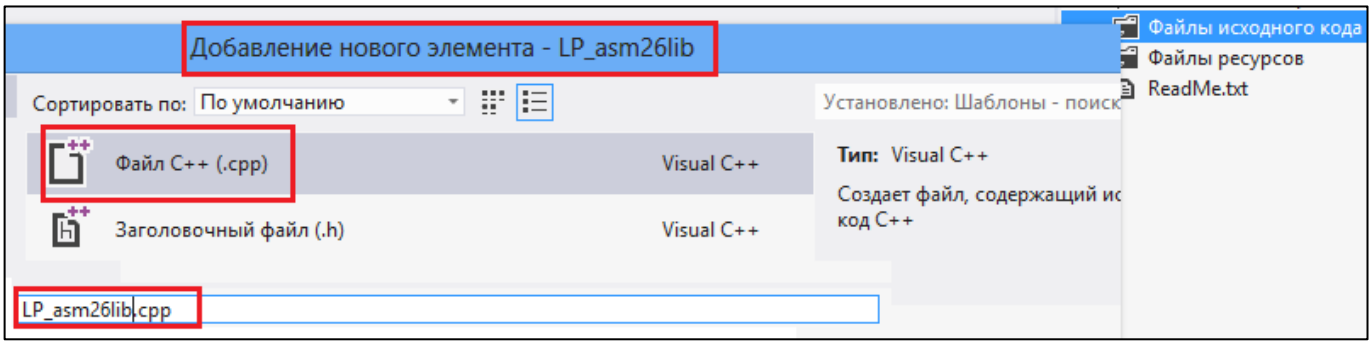


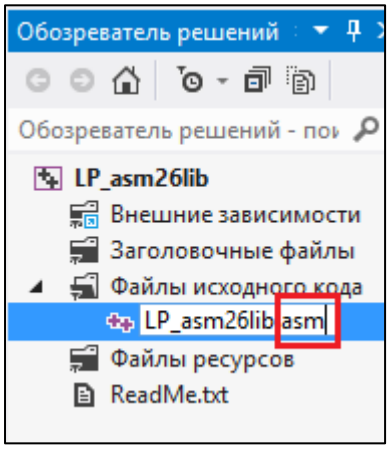
4. Создание статической библиотеки

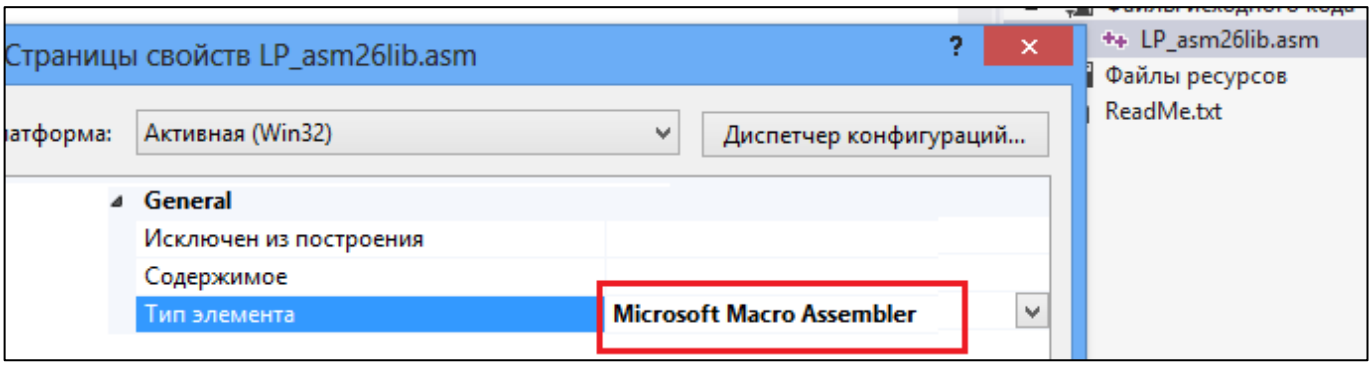


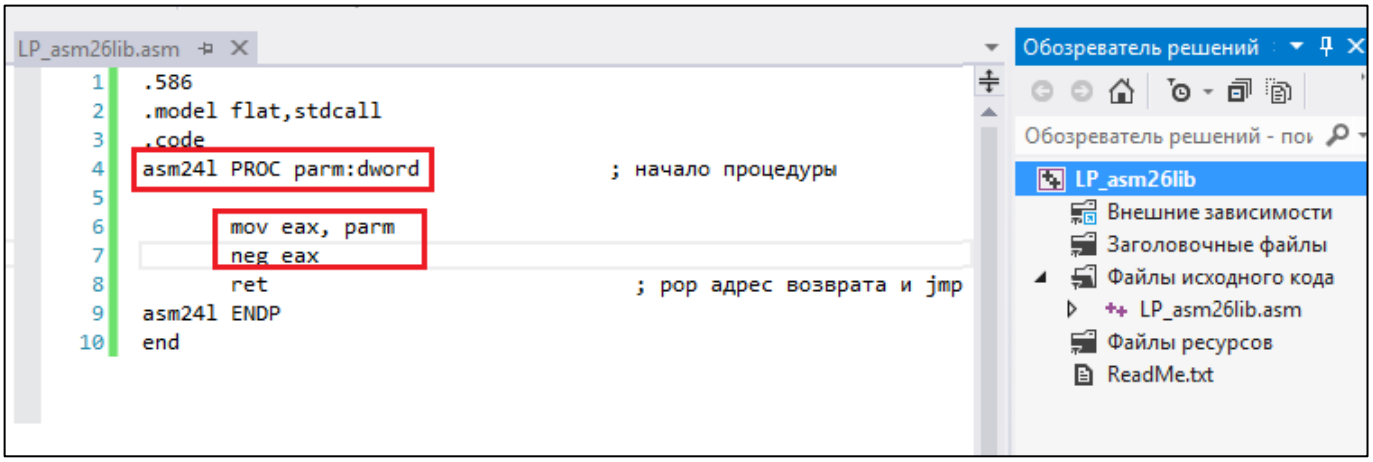


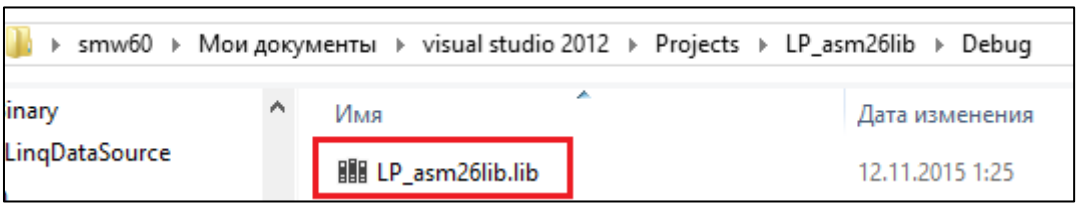

Обозреватель решений
Обозреватель решений - пои
LP_asm26lib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
Файлы ресурсов
ReadMe.txt
Файлы настройки построения для Visual C++
Доступные файлы настройки построения:
Имя
Путь
lc(.targets, .props)
$(VCTargetsPath)\BuildCustomizations\lc.targets
masm(.targets, .props)
$(VCTargetsPath)\BuildCustomizations\masm.targets

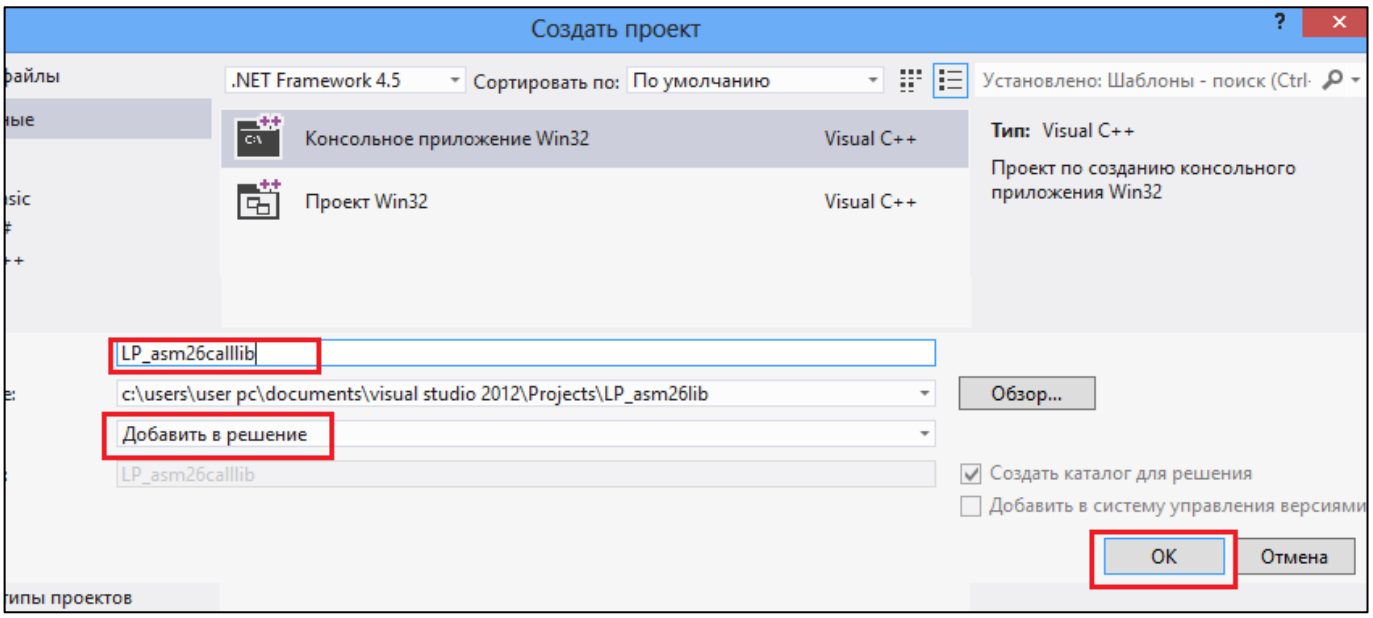

Добавление нового элемента - LP_asm26lib
Сортировать по:
По умолчанию
Установлено: Шаблоны - поиск
Файл C++ (.cpp)
Visual C++
Заголовочный файл (.h)
Visual C++
Тип: Visual C++
Создает файл, содержащий ис
код C++
LP_asm26lib.cpp
Файлы исходного кода
Файлы ресурсов
ReadMe.txt

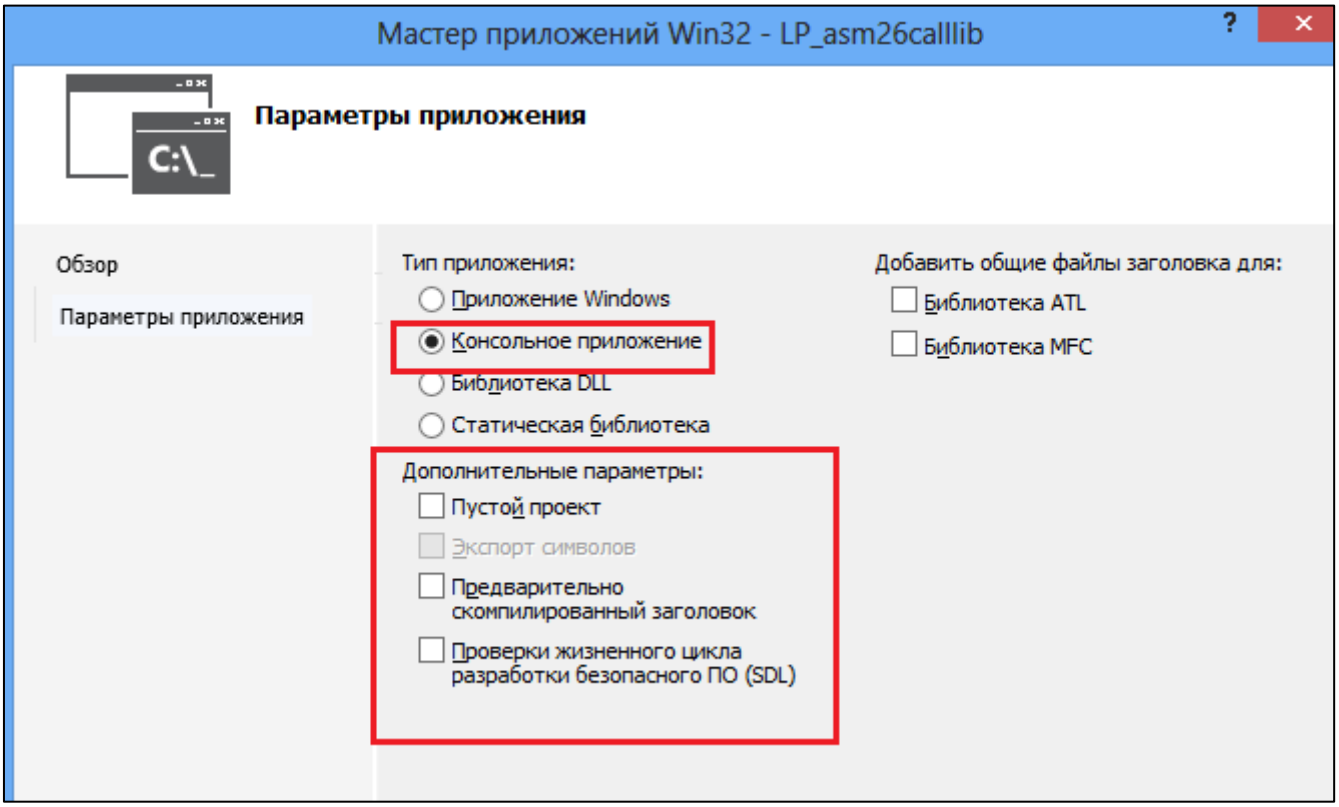

Обозреватель решений
Обозреватель решений - пои
LP_asm26lib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
LP_asm26lib.asm
Файлы ресурсов
ReadMe.txt

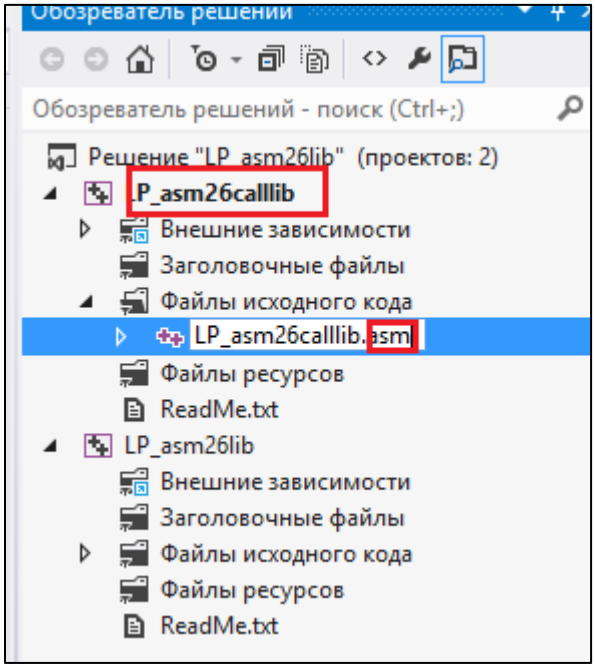

Страницы свойств LP_asm26lib.asm
атформа:
Активная (Win32)
Диспетчер конфигураций...
General
Исключен из построения
Содержимое
Тип элемента
Microsoft Macro Assembler
LP_asm26lib.asm
Файлы ресурсов
ReadMe.txt

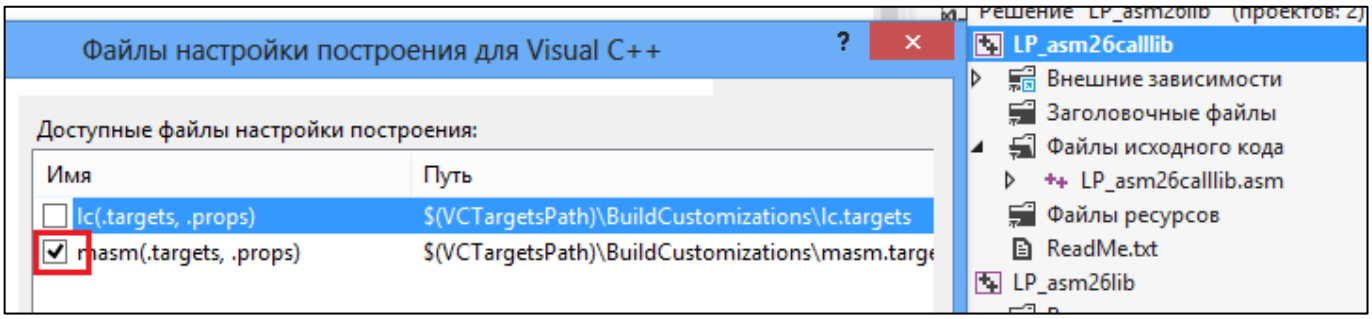

LP_asm26lib.asm
1 .586
2 .model flat,stdcall
3 .code
4 asm24l PROC parm:dword ; начало процедуры
5
6 mov eax, parm
7 neg eax
8 ret ; pop адрес возврата и jmp
9 asm24l ENDP
10 end
Обозреватель решений
Обозреватель решений - пои
LP_asm26lib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
LP_asm26lib.asm
Файлы ресурсов
ReadMe.txt

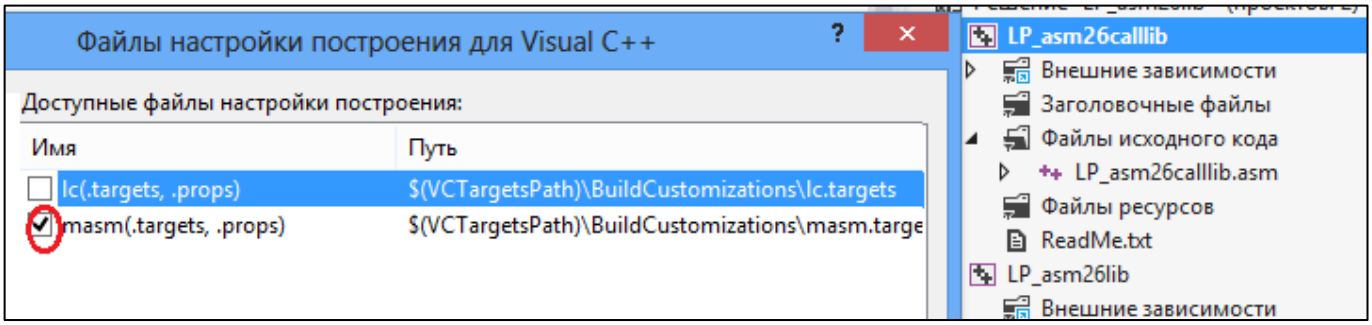

smw60 ▸ Мои документы ▸ visual studio 2012 ▸ Projects ▸ LP_asm26lib ▸ Debug
inary
LinqDataSource
Имя
Дата изменения
LP_asm26lib.lib
12.11.2015 1:25

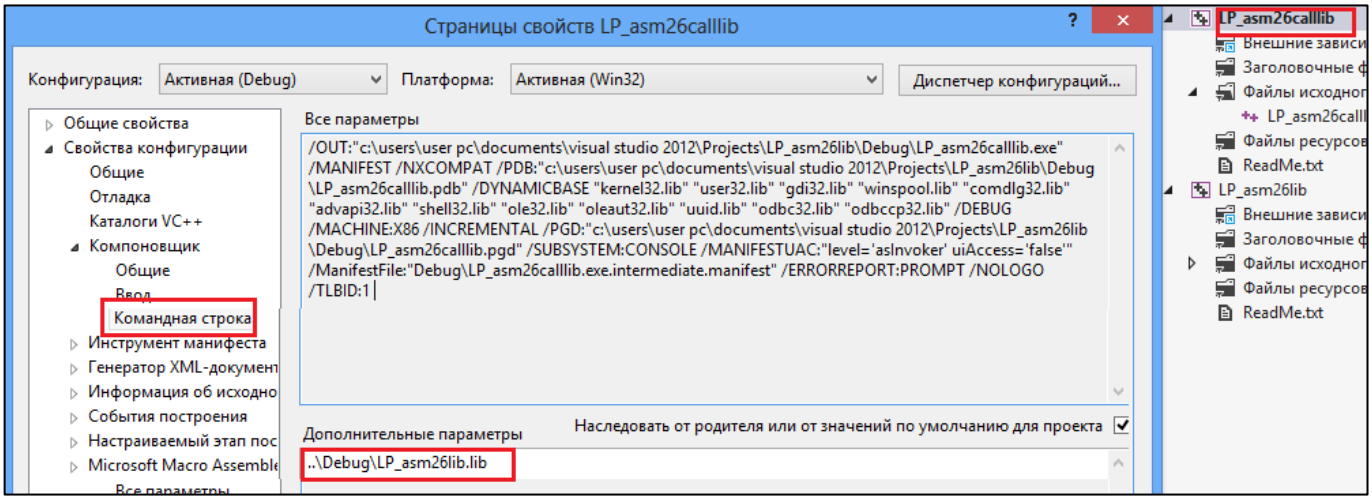

Создать проект
.NET Framework 4.5
Сортировать по: По умолчанию
Установлено: Шаблоны - поиск (Ctrl
Консольное приложение Win32 Visual C++
Проект Win32 Visual C++
Тип: Visual C++
Проект по созданию консольного приложения Win32
LP_asm26calllib
c:\users\user pc\documents\visual studio 2012\Projects\LP_asm26lib
Обзор...
Добавить в решение
LP_asm26calllib
Создать каталог для решения
Добавить в систему управления версиями
OK
Отмена

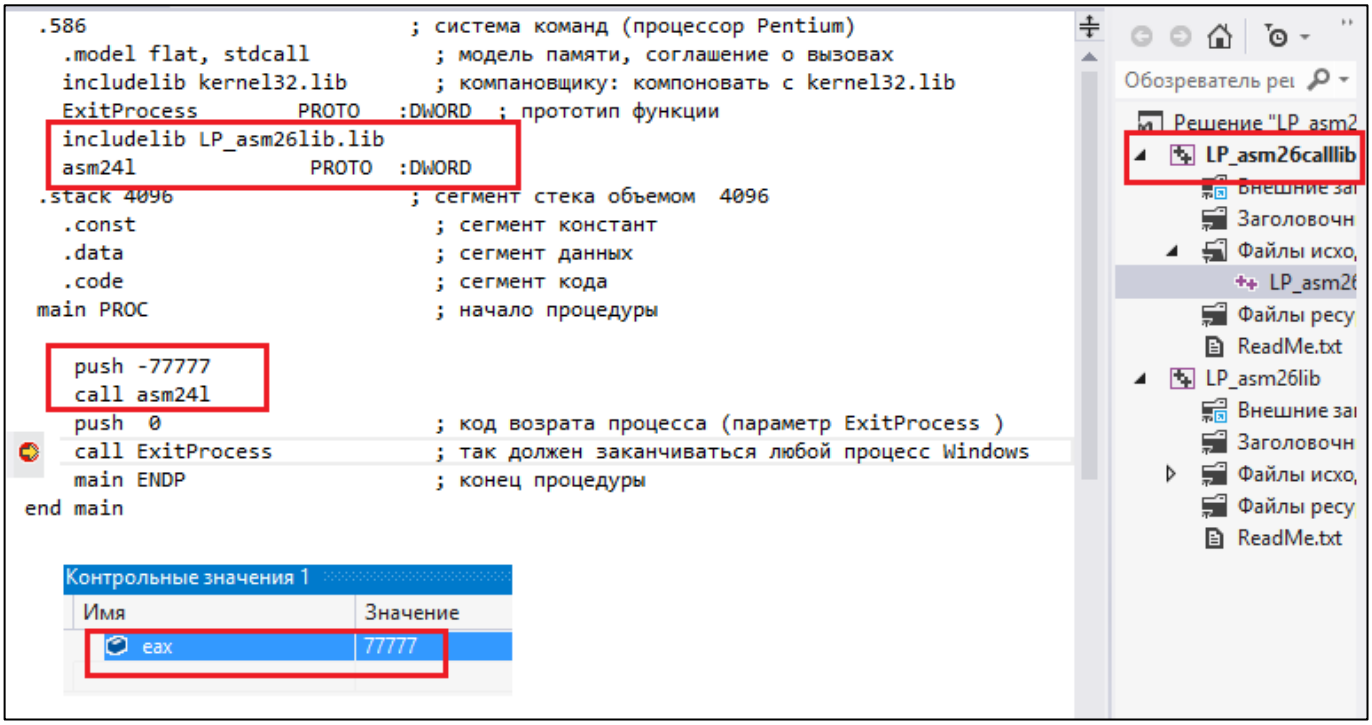

Мастер приложений Win32 - LP_asm26calllib
Параметры приложения
Обзор
Параметры приложения
Тип приложения:
Приложение Windows
Консольное приложение
Библиотека DLL
Статическая библиотека
Дополнительные параметры:
Пустой проект
Экспорт символов
Предварительно скомпилированный заголовок
Проверки жизненного цикла разработки безопасного ПО (SDL)
Добавить общие файлы заголовка для:
Библиотека ATL
Библиотека MFC


Обозреватель решений
Обозреватель решений - поиск (Ctrl+;)
Решение "LP_asm26lib" (проектов: 2)
LP_asm26calllib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
LP_asm26calllib.asm
Файлы ресурсов
ReadMe.txt
LP_asm26lib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
Файлы ресурсов
ReadMe.txt


Файлы настройки построения для Visual C++
Доступные файлы настройки построения:
Имя
Путь
lc(.targets, .props)
$(VCTargetsPath)\BuildCustomizations\lc.targets
masm(.targets, .props)
$(VCTargetsPath)\BuildCustomizations\masm.targe
LP_asm26calllib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
LP_asm26calllib.asm
Файлы ресурсов
ReadMe.txt
LP_asm26lib


Файлы настройки построения для Visual C++
Доступные файлы настройки построения:
Имя
Путь
lc(.targets, .props)
$(VCTargetsPath)\BuildCustomizations\lc.targets
masm(.targets, .props)
$(VCTargetsPath)\BuildCustomizations\masm.targe
LP_asm26calllib
Внешние зависимости
Заголовочные файлы
Файлы исходного кода
LP_asm26calllib.asm
Файлы ресурсов
ReadMe.txt
LP_asm26lib
Внешние зависимости


Страницы свойств LP_asm26calllib
Конфигурация: Активная (Debug)
Платформа: Активная (Win32)
Диспетчер конфигураций...
Общие свойства
Свойства конфигурации
Общие
Отладка
Каталоги VC++
Компоновщик
Общие
Ввод
Командная строка
Инструмент манифеста
Генератор XML-документ
Информация об исходно
События построения
Настраиваемый этап пос
Microsoft Macro Assemble
Все параметры
/OUT:"c:\users\user pc\documents\visual studio 2012\Projects\LP_asm26lib\Debug\LP_asm26calllib.exe" /MANIFEST /NXCOMPAT /PDB:"c:\users\user pc\documents\visual studio 2012\Projects\LP_asm26lib\Debug\LP_asm26calllib.pdb" /DYNAMICBASE "kernel32.lib" "user32.lib" "gdi32.lib" "winspool.lib" "comdlg32.lib" "advapi32.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "odbc32.lib" "odbccp32.lib" /DEBUG /MACHINE:X86 /INCREMENTAL /PGD:"c:\users\user pc\documents\visual studio 2012\Projects\LP_asm26lib\Debug\LP_asm26calllib.pgd" /SUBSYSTEM:CONSOLE /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /ManifestFile:"Debug\LP_asm26calllib.exe.intermediate.manifest" /ERRORREPORT:PROMPT /NOLOGO /TLBID:1
Дополнительные параметры
Наследовать от родителя или от значений по умолчанию для проекта
..\Debug\LP_asm26lib.lib
LP_asm26calllib
Внешние зависи
Заголовочные ф
Файлы исходног
LP_asm26calll
Файлы ресурсов
ReadMe.txt
LP_asm26lib
Внешние зависи
Заголовочные ф
Файлы исходног
Файлы ресурсов
ReadMe.txt


.586 ; система команд (процессор Pentium)
.model flat, stdcall ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компановщику: компоновать с kernel32.lib
ExitProcess PROTO :DWORD ; прототип функции
includelib LP_asm26lib.lib
asm24l PROTO :DWORD
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных
.code ; сегмент кода
main PROC ; начало процедуры
push -77777
call asm24l
push 0 ; код возрата процесса (параметр ExitProcess)
call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры
end main
Контрольные значения 1
Имя
Значение
eax
77777
Обозреватель ре
Решение "LP_asm2
LP_asm26calllib
Заголовочн
Файлы ресу
ReadMe.txt
LP_asm26lib
Заголовочн
Файлы ресу
ReadMe.txt

5. **Вызов функции из C++**