Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа №4

по дисциплине ЭВМ и периферийные устройства

СОЗДАНИЕ ДИНАМИЧЕСКИХ БИБЛИОТЕК

Вариант 26

Выполнил: Федоров Я.А.

студент группы ИВТ-42-23

Проверила: к.т.н.

Андреева А.А.

**Цель работы**: ознакомление использованием динамических библиотек, разработка динамической библиотеки, реализующей функции.

**Задание.** Разработать динамическую библиотеку, реализующую функции в соответствии с заданным вариантом, и программу для демонстрации ее возможностей. Использовать как явное, так и неявное связывание.

**4. function Long2Str(L: longint): string. Возвращает десятичное символьное представление числа L**.

**Код процедуры func.asm**

; Процедура Long2Str - преобразование 32-битного числа в строку

.386

.model flat

; функции, определяемые в этом DLL

public Long2Str

.code

start:

mov al, 1

ret 12

Long2Str proc

push ebp

mov ebp, esp ; Установка базового указателя

push ecx ; Сохраняем регистр CX (будем использовать как счётчик цифр)

mov edi, [ebp + 12] ; DI = адрес строки

mov eax, [ebp + 8] ; Число для преобразования

xor ecx, ecx ; ECX = 0 (счётчик цифр)

mov ebx, 10 ; EBX = 10 (делитель)

cmp eax, 0

jns skip\_sign ; Если число >= 0, пропускаем установку минуса

neg eax ; Инвертируем число (делаем его положительным)

mov edx, eax ; Сохраняем модуль числа в EDX

mov al, '-' ; Записываем знак минус

stosb ; Записываем знак в строку

mov eax, edx ; Восстанавливаем число (модуль)

skip\_sign:

cmp eax, 0

jne next\_digit ; Если EAX != 0, начинаем деление

mov al, '0' ; Если EAX == 0, просто пишем '0'

stosb

jmp finish ; Переход к завершению

next\_digit:

xor edx, edx ; Очищаем EDX перед делением

div ebx ; EAX / EBX -> результат в EAX, остаток в EDX

push edx ; Сохраняем остаток (цифру) в стек

inc ecx ; Увеличиваем счётчик цифр

cmp eax, 0

jnz next\_digit ; Повторяем, пока EAX != 0

print\_digits:

pop edx ; Достаём цифру из стека

mov al, dl

add al, '0' ; Преобразуем цифру в ASCII

stosb ; Записываем цифру в строку

loop print\_digits ; Повторяем для всех цифр

finish:

mov al, 0 ; Завершаем строку символом '$'

stosb

pop ecx ; Восстанавливаем CX

pop ebp ; Восстанавливаем BP

ret 8 ; Возвращаемся и убираем 8 байт с стека

Long2Str endp

end start

**код с неявным подключением func.dll**

; Основная программа

includelib import32.lib

includelib func.lib

extrn MessageBoxA:near

extrn ExitProcess:near

;extrn LoadLibraryA:near

;extrn FreeLibrary:near

;extrn GetProcAddress:near

MessageBox equ MessageBoxA

;LoadLibrary equ LoadLibraryA

MB\_OK equ 0000H

.386

.model flat,stdcall

.data

mas db 12 dup(?) ; строка + '$' для DOS вывода

number1 dd 2147483647 ; число для преобразования

number2 dd 12

number3 dd -2147483648

number4 dd -12

title\_str db 'result :',0

.code

extrn Long2Str:proc

start:

; Цикл по числам

lea esi, number1 ; Загружаем адрес первого числа в SI

mov ecx, 4 ; Количество чисел (4 числа)

next\_number:

push ecx

mov eax, [esi] ; Загружаем текущее число в EAX

;push eax ; Кладём число в стек

;push offset mas ; Кладём адрес буфера строки в стек

call Long2Str, eax, offset mas ; Вызов процедуры преобразования

; ret 8 внутри очистит 6 байт со стека

call MessageBox, 0, offset mas, offset title\_str, 0

; Переходим к следующему числу

add esi, 4 ; Переход к следующему числу (каждое число - 4 байта)

pop ecx

loop next\_number ; Повторяем, пока CX != 0

; Завершаем программу

call ExitProcess,0

end start

**код с явным подключением func.dll**

includelib import32.lib

extrn MessageBoxA:near

extrn ExitProcess:near

extrn LoadLibraryA:near

extrn FreeLibrary:near

extrn GetProcAddress:near

MessageBox equ MessageBoxA

.386

.model flat,stdcall

.data

mas db 12 dup(?) ; строка + '$' для DOS вывода

number1 dd 2147483647 ; число для преобразования

number2 dd 12

number3 dd -2147483648

number4 dd -12

title\_str db 'result :',0

dll\_name db 'func.dll',0

func\_name db 'Long2Str',0

hDll dd ?

pLong2Str dd ? ; сюда сохраним адрес функции

.code

start:

call LoadLibraryA, offset dll\_name

mov hDll, eax

; Получаем адрес Long2Str

call GetProcAddress, hDll, offset func\_name

mov pLong2Str, eax

; Цикл по числам

lea esi, number1 ; Загружаем адрес первого числа в SI

mov ecx, 4 ; Количество чисел (4 числа)

next\_number:

push ecx

mov eax, [esi] ; Загружаем текущее число в EAX

;push eax ; Кладём число в стек

;push offset mas ; Кладём адрес буфера строки в стек

call pLong2Str, eax, offset mas ; Вызов процедуры преобразования

; ret 8 внутри очистит 6 байт со стека

call MessageBox, 0, offset mas, offset title\_str, 0

; Переходим к следующему числу

add esi, 4 ; Переход к следующему числу (каждое число - 4 байта)

pop ecx

loop next\_number ; Повторяем, пока CX != 0

; Завершаем программу

call ExitProcess,0

end start

**содержимое секции экспорта dll файла**

File Type: DLL

Section contains the following exports for func.DLL

00000000 characteristics

0 time date stamp

0.00 version

1 ordinal base

1 number of functions

1 number of names

ordinal hint RVA name

1 0 00001005 Long2Str

Summary

1000 .edata

1000 CODE

**Содержимое секции импорта exe файла при неявном связывании**

File Type: EXECUTABLE IMAGE

Section contains the following imports:

USER32.dll

403068 Import Address Table

403050 Import Name Table

0 time date stamp

0 Index of first forwarder reference

0 MessageBoxA

KERNEL32.dll

403070 Import Address Table

403058 Import Name Table

0 time date stamp

0 Index of first forwarder reference

0 ExitProcess

func.DLL

403078 Import Address Table

403060 Import Name Table

0 time date stamp

0 Index of first forwarder reference

0 Long2Str

Summary

1000 .idata

1000 .reloc

1000 CODE

1000 DATA

**Содержимое lib файла**

р

Ђ

Long2Strs€  Long2Strfunc.DLL ы€ ы tЉ t с­ a Long2Str

Содержит информацию о том, какие функции экспортирует DLL. По сути, копия части данных из секции экспорта DLL.