


**Федеральное государственное бюджетное образовательное
учреждение высшего образования "Белгородский государственный
технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем.

Лабораторная работа № 7
Цепочечные команды.
Вариант 13

Выполнил:
Студент группы КБ-211


_____ Коренев Д.Н.

Принял:

_____ Осипов О.В.

Цель работы: изучение цепочечных команд для работы с одномерными массивами и строками.

Задание

Написать программу для решения соответствующего варианта задания. По умолчанию, если в задании не оговорено, считать что в тексте могут быть только буквы русского алфавита, латинские буквы, цифры, пробелы и нулевой символ как признак окончания строки. Пробелов между словами может быть несколько. Необходимые операции копирования, сравнения, поиска и другие организовать в виде подпрограмм, используя цепочечные команды. Если в результате преобразования длина строки увеличивается, то изначально зарезервировать для неё большой объём памяти.

Вариант	Задание
13	Преобразовать строку, внося в нее следующие исправления: если слово состоит из двух одинаковых половинок, то заключить его в угловые скобки.

```
Copyright (C) Microsoft Corp 1981-1997. All rights reserved.

Assembling: lab7.asm

*****
ASCII build
*****

Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

cat catcat doggys asas asd
cat <catcat> doggys <asas> asd
Korenev lab7 ♥ 21:44
```

```
.386
.model flat, stdcall
option casemap: none

include include\windows.inc
include include\kernel32.inc
include include\user32.inc
include include\msvcrt.inc

includelib user32.lib
includelib kernel32.lib
includelib msvcrt.lib

nline MACRO
    LOCAL nline_fmt
    .data
        nline_fmt DB 13, 10, 0
```

```

.code
    invoke crt_printf, offset nline_fmt
    add esp, 4
ENDM

.data
    dbg_hex64_fmt DB "DBG: 0x%016lX", 13, 10, 0
    dbg_hex32_fmt DB "DBG: 0x%08lX", 13, 10, 0
    dbg_hex16_fmt DB "DBG: 0x%04X", 13, 10, 0
    dbg_hex8_fmt DB "DBG: 0x%02X", 13, 10, 0

    dbg_int64_fmt DB "DBG: 0d%lld", 13, 10, 0
    dbg_int32_fmt DB "DBG: 0d%d", 13, 10, 0
    dbg_int16_fmt DB "DBG: 0d%hd", 13, 10, 0
    dbg_int8_fmt DB "DBG: 0d%hhd", 13, 10, 0

    dbg_float64_fmt DB "DBG: 0f%lf", 13, 10, 0
    dbg_float32_fmt DB "DBG: 0f%f", 13, 10, 0

    dbg_nline_fmt DB 13, 10, 0
.code

; =====
; uint32_t get_str_size(char* str_ptr);
; Возвращает размер строки до символа "\0"
; =====
get_str_size PROC str_ptr:DWORD
    LOCAL _size:DWORD

    PUSH EDX
    PUSH ECX

    MOV EAX, str_ptr
    MOV ECX, 0
    MOV _size, 0
loop_get_str_size:
    MOV DL, [EAX+ECX]
    CMP DL, 0
    JE end_loop_get_str_size
    INC _size
    INC ECX
    JMP loop_get_str_size

end_loop_get_str_size:
    MOV EAX, _size

    POP ECX
    POP EDX

    RET 4

```

```

get_str_size ENDP

; =====
; uint32_t get_first_word_size(char* str_ptr);
; Возвращает размер строки до символа '\0' или ' '
; =====
get_first_word_size PROC str_ptr:DWORD
    LOCAL _size:DWORD

    PUSH EDX
    PUSH ECX

    MOV EAX, str_ptr
    MOV ECX, 0
    MOV _size, 0
loop_get_first_word_size:
    MOV DL, [EAX+ECX]
    CMP DL, 32
    JE end_loop_get_first_word_size
    CMP DL, 0
    JE end_loop_get_first_word_size

    INC _size
    INC ECX
    JMP loop_get_first_word_size

end_loop_get_first_word_size:
    MOV EAX, _size

    POP ECX
    POP EDX

    RET 4
get_first_word_size ENDP

; =====
; int32_t is_doubleword(char* str, char* res_str);
; Возвращает 1, если слово состоит из двух одинаковых половинок
; =====
is_doubleword PROC str_ptr:DWORD

    PUSH ESI
    PUSH EDI
    PUSH EBX
    PUSH ECX
    PUSH EDX

    invoke get_str_size, str_ptr
    ; Проверить на четность
    MOV EDI, EAX
    ; Размер строки size

```

```

    AND EAX, 1
    CMP EAX, 1
    JE __is_doubelword_odd
    JNE __is_doubelword_even

__is_doubelword_odd:
    MOV EAX, 0
    JMP __is_doubelword_end

__is_doubelword_even:
    XOR EDX, EDX
    MOV EAX, EDI
    MOV EBX, 2
    DIV EBX                ; EAX = size/2

    MOV ESI, str_ptr       ; ESI = str_ptr
    MOV EDI, str_ptr
    ADD EDI, EAX            ; EDI = str_ptr + size/2

    MOV ECX, EAX
    REPE CMPSB              ; Сравнить первую половину со второй
    JA __is_doubelword_not
    JB __is_doubelword_not
    MOV EAX, 1
    JMP __is_doubelword_end

__is_doubelword_not:
    MOV EAX, 0

__is_doubelword_end:

    POP EDX
    POP ECX
    POP EBX
    POP EDI
    POP ESI

    ret 4
is_doubelword ENDP

; =====
; void get_n_word(char* str_ptr, char* res_str_ptr, uint32_t n);
; Помещает в res_str_ptr n-ную подстроку строки str_ptr.
; Подстроки разделены символом ' '
; =====
get_n_word PROC str_ptr:DWORD, res_str_ptr:DWORD, n:DWORD
    LOCAL _str_ptr:DWORD
    LOCAL _str_size:DWORD
    LOCAL _i:DWORD

```

```

    PUSH ESI
    PUSH EDI
    PUSH EBX
    PUSH ECX

    MOV _i, 0
    mov EAX, str_ptr           ; Получить адрес строки
    mov _str_ptr, EAX         ; _str_ptr = str_ptr

__loop:
    invoke get_first_word_size, _str_ptr    ; EAX = size of str_ptr
    MOV _str_size, EAX                     ; _str_size = size of str_ptr

    MOV ECX, _str_size                     ; ECX = size of str_ptr
    MOV ESI, _str_ptr                       ; ESI = str_ptr
    MOV EDI, res_str_ptr                   ; EDI = res_str_ptr
    REP MOVSB                               ; Скопировать строку в res_str_ptr

    MOV EAX, _str_ptr                       ; EAX = str_ptr
    add EAX, _str_size                       ; _str_ptr += size of str_ptr
    inc EAX
    mov _str_ptr, EAX                       ; _str_ptr = str_ptr
    inc _i
    mov EAX, _i
    mov EBX, n
    cmp EAX, EBX
    jle __loop

    mov EAX, res_str_ptr
    mov EBX, _str_size
    mov BYTE PTR [EAX+EBX], 0

    POP ECX
    POP EBX
    POP EDI
    POP ESI

    RET 12
get_n_word ENDP

; =====
; void wrap_corners(char* str_ptr, char* res_str_ptr);
; =====
wrap_corners PROC str_ptr:DWORD, res_str_ptr:DWORD
    LOCAL _str_ptr:DWORD
    LOCAL _res_str_ptr:DWORD
    LOCAL _str_size:DWORD

    PUSH ESI
    PUSH EDI

```

```

PUSH EAX
PUSH ECX

mov EAX, str_ptr          ; Получить адрес строки
mov _str_ptr, EAX         ; _str_ptr = str_ptr

mov EAX, res_str_ptr      ; Получить адрес строки
mov _res_str_ptr, EAX     ; _res_str_ptr = res_str_ptr

mov EAX, _res_str_ptr
mov BYTE PTR [EAX], '<'
inc EAX
mov _res_str_ptr, EAX

invoke get_str_size, _str_ptr ; EAX = size of str_ptr
MOV _str_size, EAX           ; _str_size = size of str_ptr

MOV ECX, _str_size           ; ECX = size of str_ptr
MOV ESI, _str_ptr            ; ESI = str_ptr
MOV EDI, _res_str_ptr        ; EDI = res_str_ptr
REP MOVSB                    ; Скопировать строку в res_str_ptr

mov EAX, _res_str_ptr
add EAX, _str_size
mov BYTE PTR [EAX], '>'
inc EAX
mov BYTE PTR [EAX], 0

POP ECX
POP EAX
POP EDI
POP ESI

ret 8
wrap_corners ENDP

; =====
; uint32_t count_words(char* str_ptr);
; Возвращает количество слов в строке str_ptr
; =====
count_words PROC str_ptr:DWORD
    LOCAL _str_ptr:DWORD
    LOCAL _str_size:DWORD
    LOCAL _cnt:DWORD

    PUSH EBX

    MOV _cnt, 1
    mov EAX, str_ptr          ; Получить адрес строки
    mov _str_ptr, EAX         ; _str_ptr = str_ptr

```

```

        invoke get_str_size, _str_ptr
        MOV _str_size, EAX                                ; _str_size = size of str_ptr
_loop:
        mov EAX, _str_ptr
        movsx EAX, BYTE PTR [EAX]
        mov EBX, ' '
        cmp EAX, EBX
        je _inc_cnt
        jne _pass

_inc_cnt:
        add _cnt, 1
_pass:

        dec _str_size
        inc _str_ptr
        cmp _str_size, 0
        jne _loop

        mov EAX, _cnt

        POP EBX

        RET 4
count_words ENDP

; =====
; void transform_str(char* str_ptr, char* res_str_ptr);
; Если в строке есть слово, состоящее из двух одинаковых половинок, то
; поместить это слово в кавычки.
; =====
transform_str PROC str_ptr:DWORD, res_str_ptr:DWORD
    LOCAL _str_ptr:DWORD
    LOCAL _res_str_ptr:DWORD
    LOCAL _str_size:DWORD
    LOCAL _i:DWORD
    LOCAL _cnt:DWORD
    LOCAL _temp_size:DWORD
.data
    __transform_str__buffer db 1024 DUP(0)
    __transform_str__buffer1 db 1024 DUP(0)
.code

    MOV _i, 0

    mov EAX, str_ptr                                ; Получить адрес строки
    mov _str_ptr, EAX                                ; _str_ptr = str_ptr

    mov EAX, res_str_ptr                            ; Получить адрес строки

```



```

mov _res_str_ptr, EAX                ; _res_str_ptr = res_str_ptr

invoke get_str_size, _str_ptr        ; EAX = size of str_ptr
MOV _str_size, EAX                  ; _str_size = size of str_ptr

invoke count_words, _str_ptr
mov _cnt, EAX

__loop:
    invoke get_n_word, _str_ptr, offset __transform_str__buffer, _i

    invoke is_doubelword, offset __transform_str__buffer

    cmp EAX, 1
    je __wrap
    jne __glue

__wrap:
    invoke wrap_corners, offset __transform_str__buffer, offset
__transform_str__buffer1

    mov ESI, offset __transform_str__buffer1
    mov EDI, offset __transform_str__buffer
    mov ECX, 1024
    rep movsb

__glue:
    invoke get_str_size, offset __transform_str__buffer
    mov _temp_size, EAX              ; _temp_size = size of buffer

    mov ESI, offset __transform_str__buffer
    mov EDI, _res_str_ptr
    mov ECX, _temp_size
    rep movsb

    mov EAX, _res_str_ptr
    add EAX, _temp_size
    mov BYTE PTR [EAX], ' '
    inc EAX
    mov _res_str_ptr, EAX

    inc _i
    dec _cnt
    cmp _cnt, 0
    jne __loop

    mov EAX, _res_str_ptr
    mov BYTE PTR [EAX-1], 0

ret 8

```

```

transform_str ENDP

start:
.data
    strr db "cat catcat doggys asas asd", 0
    res_str db "#", 1024 DUP(?), 0
.code
    invoke crt_printf, offset strr
    invoke crt_printf, offset dbg_nline_fmt
    invoke transform_str, offset strr, offset res_str
    invoke crt_printf, offset res_str

    invoke ExitProcess, 0 ; Выход из программы
end start

```

Вывод: в ходе данной лабораторной работы мы изучили цепочечные команды для работы с одномерными массивами и строками.