**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:

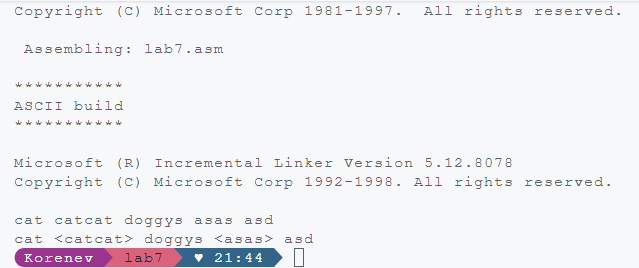
Осипов О.В.

*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, внеся в нее следующие исправления: если слово состоит из двух одинаковых половинок, то заключить его в угловые скобки. |



.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%016llX", 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\_doubelword 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 изученили цепочечные команды для работы с одномерными массивами и строками.