

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

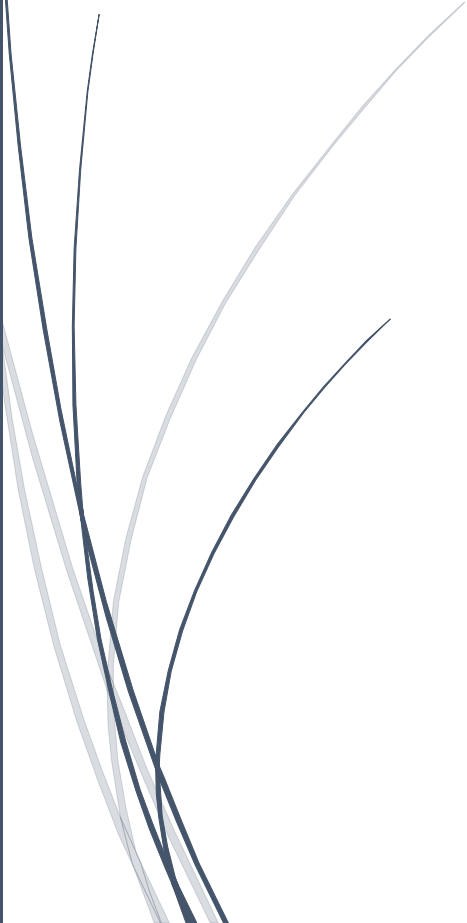
27.10.2020

Пояснительная записка

Микропроект по ABC

НИУ ВШЭ

Профессор департамента
программной инженерии
факультета компьютерных наук
Легалов Александр Иванович

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Бакытбек уулу Нуржигит -
БПИ197, ВАРИАНТ 3

Задание

Разработать программу, которая меняет на обратный порядок следования символов каждого слова в ASCII-строке символов

Описание проекта

Проект разработан на языке Ассемблер. Пользователю предлагается ввести строку, после ввода выводится строка в обратном порядке. Если длина строки больше или равно 1024 выводится сообщение, что длина должна быть меньше 1024.

Основные алгоритмы

Имеются два основных алгоритма – подсчет длины строки и переворачивание строки.

Первый проходится по символам строки, увеличивая счетчик по модулю, пока не встретит символ конца строки - "0".

Второй если длина меньше или равно 1, то ничего не делает, если больше, то меняет первый с последним, второй с предпоследним ...

Пример выполнения программы

Текст программы

```
; Bakytbek uulu Nurzhigit 197, var #3
; Task:
; Develop a program that changes to reverse character order
; each word in an ASCII character string

format PE console
entry start

include 'win32a.inc'

;-----
; Data section
section '.data' data readable writable

msgEnter db      'Type text and press Enter ;)', 10, 0
msgRev  db      'Reversed string :D',10, 0
msgSizeF db      'Size of text should be less than 1024. Pleae try again :(',10, 0
fString db      '%s',0
text    db      'Hello! My algorithm reverses the string :)',0
strB    rb      1200
em      db      '',10,0
;-----
; Main program
section '.code' code readable executable
start:
; Reverse text

    invoke printf, text                ; print to console text
    add esp, 4

    invoke printf, em                  ; print to console empty text
    add esp, 4

    stdcall StrLen,text                ; calculate text size
    stdcall StrRev, text, ecx          ; reverse the text

    invoke printf, text                ; print to console text
    add esp, 4

    invoke printf, em                  ; print to console text
    add esp, 4
;-----
```

```

; Reverse input word
    invoke printf, msgEnter          ; print to console "Type text"
    add esp, 4

    invoke gets, strB                ; read from console to text
    add esp, 4

    stdcall StrLen, strB              ; calculate string's size
    cmp ecx, 1024
    jge failSize                     ; go to failSize if size of text >= 1024

    stdcall StrRev, strB, ecx         ; reverse the string

    invoke printf, msgRev             ; print to console "Reversed word"
    add esp, 4
    invoke printf, strB               ; print reversed string
    add esp, 4
finish:
    invoke getch                     ; stop console closing
    invoke ExitProcess, 0             ; exit

```

;-----

```

failSize:
    invoke printf, msgSizeF          ; print to console "Fail size"
    add esp, 4
    invoke getch                     ; stop console closing
    invoke ExitProcess, 0

```

; reverse string

```

proc StrRev strAddr, strSize
    push esi                        ; save the values of the used registers
    push edi
    mov esi, [strAddr]              ; set esi the address of the beginning of the string
    mov eax, [strSize]               ; eax = strSize
    lea edi, [esi+eax]               ; set edi the address of the end of the string
    cmp eax, 2                       ; compare eax and 2
    jnb StrRevExit                   ; eax < 2 (c-strings ends with 0)

```

Alg:

```

    sub edi, 1                       ; shift end "iterator" to the left
    mov dh, [edi]                    ; save value of end and right "iterator" to dh
    mov dl, [esi]                    ; save value of begin and right "iterator" to dl

```

```

        mov [esi],dh        ; swap value of begin
        mov [edi],dl        ; and end "iterator"
        add esi,1           ; shift begin "iterator" to the right
        cmp esi,edi         ; | if begin < end go
        jb Alg              ; | to Alg label
StrRevExit:
        pop edi             ; Restoring the values of the used registers
        pop esi
        ret
endp

;-----
;calculate length of string
proc StrLen, strInput
    mov ecx,-1
    xor     al, al          ; tail symbol is zero
    mov edi,[strInput]     ; using strInput as stack argument
    cld                    ; direction from begin to end
    repne scasb             ; while(strInput[edi] != al) {edi++; ecx--;}
    not ecx                 ; | does the same as neg but
    dec ecx                 ; | doesn't change the flags
    ret
endp

;-----
; Including External Api
section '.idata' import data readable
    library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll'

include 'api\kernel32.inc'
import kernel,\
    ExitProcess, 'ExitProcess'
import msvcrt,\
    printf, 'printf',\
    scanf, 'scanf',\
    getch, '_getch',\
    gets, 'gets'

```