**Комп’ютерний практикум №4**

**Тема:** масиви.

**Завдання**:

Скласти програму на нижче наведені завдання:

1. Написати програму знаходження суми елементів одномірного масиву, елементи вводить користувач.
2. Написати програму пошуку максимального (або мінімального) елемента одномірного масиву, елементи вводить користувач.
3. Написати програму сортування одномірного масиву цілих чисел загального вигляду.
4. Написати програму пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий вводить користувач.

**Код програми**:

; STACK SEGMENT

STSEG SEGMENT PARA STACK "STACK"

    DB 64 DUP (0)

STSEG ENDS

; DATA SEGMENT

DSEG SEGMENT PARA PUBLIC "DATA"

    helloStr db "Enter number of elements (1-20) : $"

    elementStr db "Enter an element: $"

    overflowError db "Overflow error", 10, "$"

    numberOfElementsError db "Invalid number!", 10, "$"

    elementError db "String can not be empty"

    findElemStr db "Enter an element to find : $"

    xStr db "Enter X (1-5): $"

    yStr db "Enter Y (1-5): $"

    arrayStr db "Array is : $"

    sumStr db "Sum is : $"

    minStr db "Min is : $"

    sortedStr db "Sorted array is : $"

    buffer db 6, ?, 6 dup ('?')

    xBuffer db 6, ?, 6 dup ('?')

    yBuffer db 6, ?, 6 dup ('?')

    x dw ?

    y dw ?

    elements dw 20 dup (?)

    twoDArr dw 25 dup (?)

    len dw 0

    i dw 0

    j dw 0

    tmp dw 0

    sum dw 0

    min dw ?

    find dw ?

    was\_overflow db 0

    was\_empty db 0

DSEG ENDS

; CODE SEGMENT

CSEG SEGMENT PARA PUBLIC "CODE"

    MAIN PROC FAR

        ASSUME cs: CSEG, ds: DSEG, ss:STSEG

        push ds

        xor ax, ax

        push ax

        mov ax, DSEG

        mov ds, ax

        lea dx, helloStr

        call far ptr WRITING

        lea dx, buffer

        call far ptr READING

        mov ax, 10

        int 29h

        lea bx, buffer + 2

        call far ptr ATOI

        cmp was\_overflow, 1

        je jump\_toEnd

        mov len, ax

        cmp len, 1

        jl jump\_toError

        cmp len, 20

        jg jump\_toError

        mov ax, len

        mov i, ax

        reading\_loop:

            lea dx, elementStr

            call far ptr WRITING

            lea dx, buffer

            call far ptr READING

            mov ax, 10

            int 29h

            lea bx, buffer + 2

            call far ptr ATOI

            cmp was\_overflow, 1

            je jump\_toEnd

            mov bx, len

            sub bx, i

            shl bx, 1

            mov elements[bx], ax

            dec i

            jnz reading\_loop

        lea dx, arrayStr

        call far ptr WRITING

        mov ax, len

        mov i, ax

        jmp output\_array

        jump\_toError: jmp error\_main

        jump\_toEnd: jmp end\_main

        output\_array:

            mov bx, len

            sub bx, i

            shl bx, 1

            mov ax, elements[bx]

            mov bx, ax

            call far ptr ITOA

            mov ax, ' '

            int 29h

            dec i

            jnz output\_array

        mov ax, 10

        int 29h

        lea dx, sumStr

        call far ptr WRITING

        call far ptr ARRAY\_SUM

        cmp was\_overflow, 1

        je jump\_toEnd4

        mov bx, sum

        call far ptr ITOA

        mov ax, 10

        int 29h

        lea dx, minStr

        call far ptr WRITING

        call far ptr ARRAY\_MIN

        mov bx, min

        call far ptr ITOA

        mov ax, 10

        int 29h

        lea dx, sortedStr

        call far ptr WRITING

        call far ptr ARRAY\_SORT

        mov ax, len

        mov i, ax

        jmp output\_array1

        jump\_toEnd4: jmp jump\_toEnd3

        output\_array1:

            mov bx, len

            sub bx, i

            shl bx, 1

            mov ax, elements[bx]

            mov bx, ax

            call far ptr ITOA

            mov ax, ' '

            int 29h

            dec i

            jnz output\_array1

        mov ax, 10

        int 29h

        ; READ X

        lea dx, xStr

        call far ptr WRITING

        lea dx, xBuffer

        call far ptr READING

        mov ax, 10

        int 29h

        ; READ Y

        lea dx, yStr

        call far ptr WRITING

        lea dx, yBuffer

        call far ptr READING

        mov ax, 10

        int 29h

        lea bx, xBuffer + 2

        call far ptr ATOI

        cmp was\_overflow, 1

        je jump\_toEnd3

        mov x, ax

        lea bx, yBuffer + 2

        call far ptr ATOI

        cmp was\_overflow, 1

        je jump\_toError3

        mov y, ax

        cmp x, 1

        jl jump\_toError3

        cmp x, 5

        jg jump\_toError3

        cmp y, 1

        jl jump\_toError3

        cmp y, 5

        jg jump\_toError3

        mov i, 0

        jmp reading\_2d\_out

        jump\_toEnd3: jmp jump\_toEnd2

        jump\_toError3: jmp jump\_toError2

        reading\_2d\_out:

            mov j, 0

            reading\_2d\_in:

                mov ax, '['

                int 29h

                mov bx, i

                call far ptr ITOA

                mov ax, ','

                int 29h

                mov bx, j

                call far ptr ITOA

                mov ax, ']'

                int 29h

                mov ax, ' '

                int 29h

                lea dx, buffer

                call far ptr READING

                mov ax, 10

                int 29h

                lea bx, buffer + 2

                call far ptr ATOI

                cmp was\_overflow, 1

                je jump\_toEnd2

                mov tmp, ax

                mov ax, i

                mov bx, x

                mul bx

                add ax, j

                mov bx, ax

                shl bx, 1

                mov ax, tmp

                mov twoDArr[bx], ax

                inc j

                mov ax, j

                cmp ax, x

                jl reading\_2d\_in

            inc i

            mov ax, i

            cmp ax, y

            jl reading\_2d\_out

        mov i, 0

        jmp writing\_2d\_out

        jump\_toEnd2: jmp end\_main

        jump\_toError2: jmp error\_main

        writing\_2d\_out:

            mov j, 0

            writing\_2d\_in:

                mov ax, i

                mov bx, x

                mul bx

                add ax, j

                mov bx, ax

                shl bx, 1

                mov ax, twoDArr[bx]

                mov bx, ax

                call far ptr ITOA

                mov ax, ' '

                int 29h

                inc j

                mov ax, j

                cmp ax, x

                jl writing\_2d\_in

            mov ax, 10

            int 29h

            inc i

            mov ax, i

            cmp ax, y

            jl writing\_2d\_out

        lea dx, findElemStr

        call far ptr WRITING

        lea dx, buffer

        call far ptr READING

        lea bx, buffer + 2

        call far ptr ATOI

        cmp was\_overflow, 1

        je end\_main

        mov find, ax

        call far ptr FIND\_ELEM

        jmp end\_main

        error\_main:

            lea dx, numberOfElementsError

            call far ptr WRITING

        end\_main:

            ret

    MAIN ENDP

    FIND\_ELEM PROC FAR

        mov ax, 10

        int 29h

        mov i, 0

        find\_2d\_out:

            mov j, 0

            find\_2d\_in:

                mov ax, i

                mov bx, x

                mul bx

                add ax, j

                mov bx, ax

                shl bx, 1

                mov ax, twoDArr[bx]

                cmp ax, find

                jne find\_skip

                mov ax, '['

                int 29h

                mov bx, i

                call far ptr ITOA

                mov ax, ','

                int 29h

                mov bx, j

                call far ptr ITOA

                mov ax, ']'

                int 29h

                mov ax, 10

                int 29h

                find\_skip:

                    inc j

                    mov ax, j

                    cmp ax, x

                    jl find\_2d\_in

            inc i

            mov ax, i

            cmp ax, y

            jl find\_2d\_out

        ret

    ENDP FIND\_ELEM

    ARRAY\_SORT PROC FAR

        mov i, 0

        out\_loop\_sort:

        mov ax, i

        mov min, ax

        mov j, ax

        inc j

        mov ax, len

        cmp j, ax

        jge sort\_end

        inner\_loop\_sort:

            mov bx, min

            shl bx, 1

            mov ax, elements[bx]

            mov bx, j

            shl bx, 1

            mov cx, elements[bx]

            cmp ax, cx

            jng skip\_update

            mov ax, j

            mov min, ax

            skip\_update:

            inc j

            mov ax, j

            cmp ax, len

            jl inner\_loop\_sort

        mov bx, i

        shl bx, 1

        mov ax, elements[bx]

        mov bx, min

        shl bx, 1

        mov cx, elements[bx]

        mov dx, cx

        mov elements[bx], ax

        mov bx, i

        shl bx, 1

        mov elements[bx], dx

        inc i

        mov ax, len

        cmp i, ax

        jl out\_loop\_sort

        sort\_end:

            ret

    ENDP ARRAY\_SORT

    ARRAY\_SUM PROC FAR

        mov ax, len

        mov i, ax

        sum\_loop:

            mov bx, len

            sub bx, i

            shl bx, 1

            mov ax, elements[bx]

            add sum, ax

            jo error\_sum

            dec i

            jnz sum\_loop

        jmp sum\_end

        error\_sum:

            mov was\_overflow, 1

            lea dx, overflowError

            call far ptr WRITING

        sum\_end:

            ret

    ENDP ARRAY\_SUM

    ARRAY\_MIN PROC FAR

        mov ax, elements[0]

        mov min, ax

        mov ax, len

        mov i, ax

        min\_loop:

            mov bx, len

            sub bx, i

            shl bx, 1

            mov ax, elements[bx]

            cmp ax, min

            jg iterate\_min

            mov min, ax

            iterate\_min:

                dec i

                jnz min\_loop

        ret

    ENDP ARRAY\_MIN

    ITOA PROC FAR

        or bx, bx

        jns positive\_number

        xor ax, ax

        mov al, '-'

        int 29h

        neg bx

        positive\_number:

            mov ax, bx

            xor cx, cx  ; chars number

            mov bx, 10  ; diviator

            itoa\_loop:

                xor dx, dx  ; remainder stores here

                div bx

                add dl, '0'

                push dx

                inc cx

                test ax, ax

                jnz itoa\_loop

            output:

                pop ax

                int 29h

                loop output

            ret

        ITOA ENDP

    ATOI PROC FAR

        xor ax, ax      ; result

        xor cx, cx      ; char

        xor di, di      ; 10

        mov di, 10

        xor si, si      ; sign

        skip\_whitespaces:

            mov cl, BYTE PTR [bx]

            cmp cl, 32  ; ' '

            je step

            cmp cl , 9  ; '\t'

            je step

            jmp sign\_check

        step:

            inc bx

            jmp skip\_whitespaces

        sign\_check:

            cmp cl, 45  ; '-'

            je minus

            cmp cl, 43  ; '+'

            je plus

            jmp atoi\_loop

        minus:

            mov si, 1

            plus:

                inc bx

                jmp atoi\_loop

        atoi\_loop:

            mov cl, BYTE PTR [bx]

            cmp cl, 48  ; '0'

            jl atoi\_end

            cmp cl, 57  ; '9'

            jg atoi\_end

            imul di

            jo error

            sub cl, 48

            add ax, cx

            jo error

            inc bx

            jmp atoi\_loop

        atoi\_end:

            cmp si, 1

            je make\_neg

            jmp exit\_atoi

        make\_neg:

            neg ax

            jmp exit\_atoi

        error:

            mov was\_overflow, 1

            xor dx, dx

            lea dx, overflowError

            call far ptr WRITING

        exit\_atoi:

            ret

    ATOI ENDP

    READING PROC FAR

        mov ah, 10

        int 21h

        ret

    READING ENDP

    WRITING PROC FAR

        mov ah, 9

        int 21h

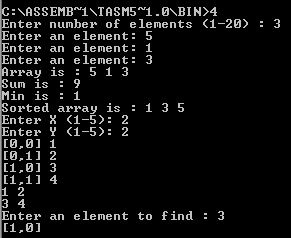
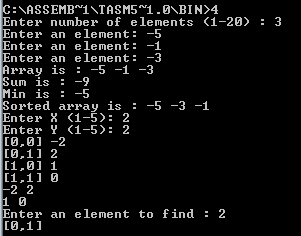
        ret

    WRITING ENDP

CSEG ENDS

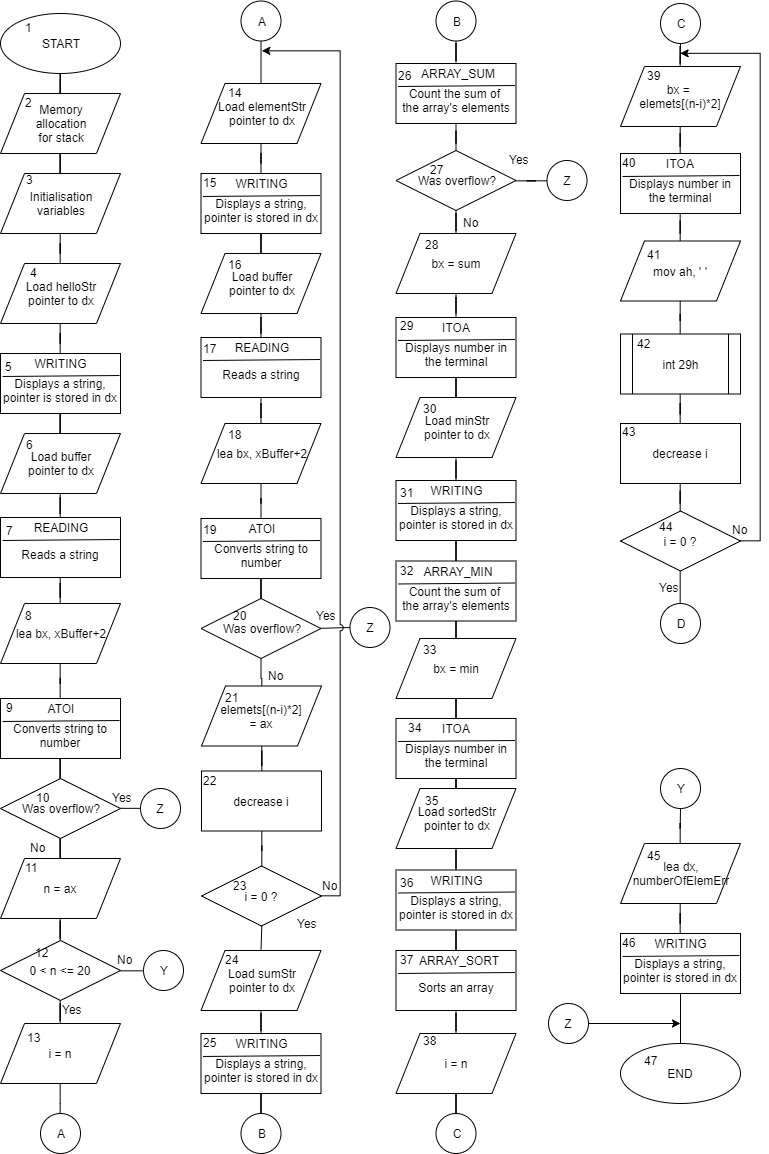
END MAIN

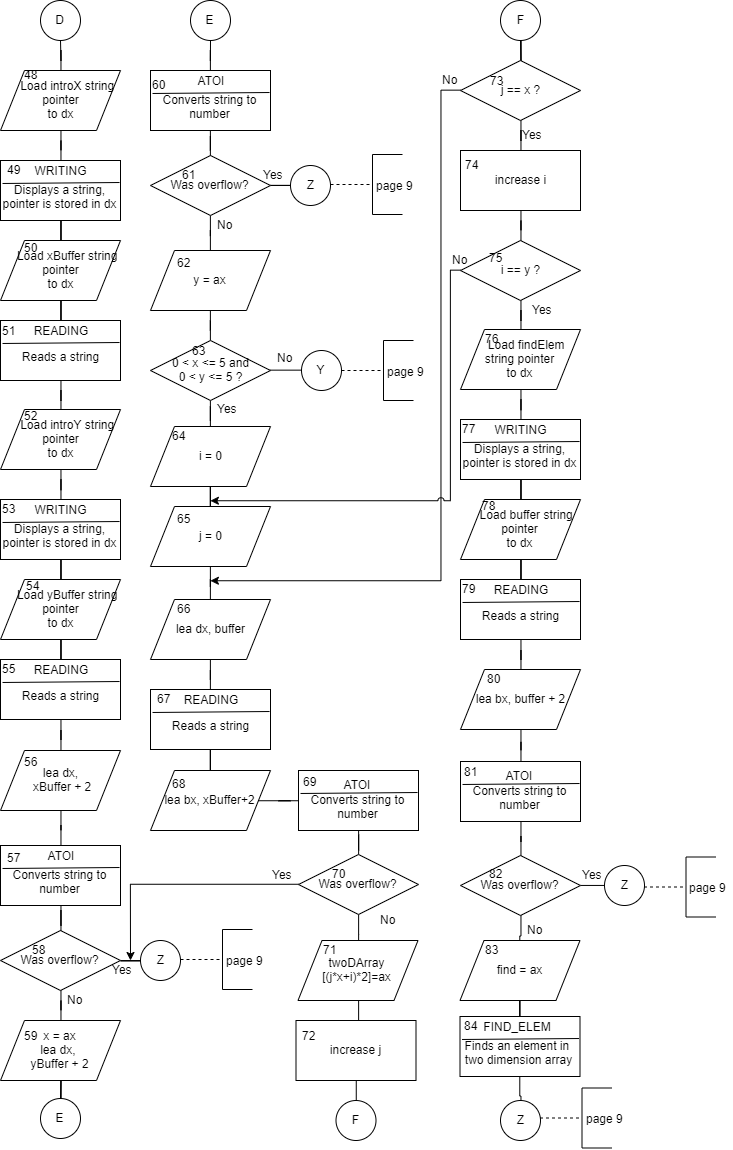
**Скріншоти:**

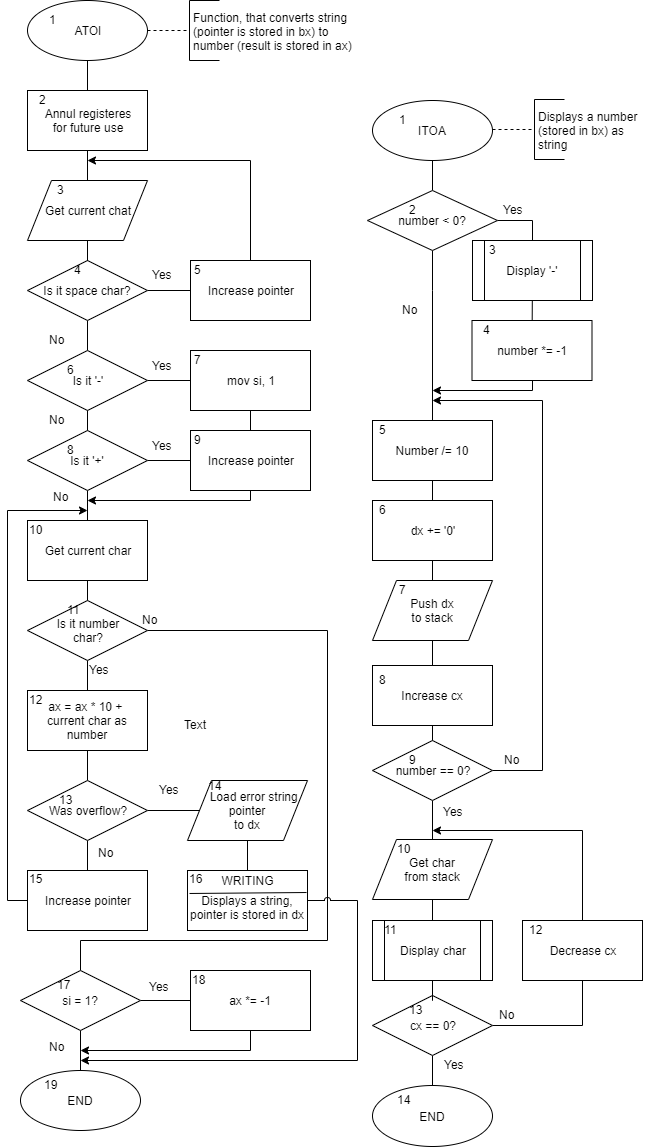
****

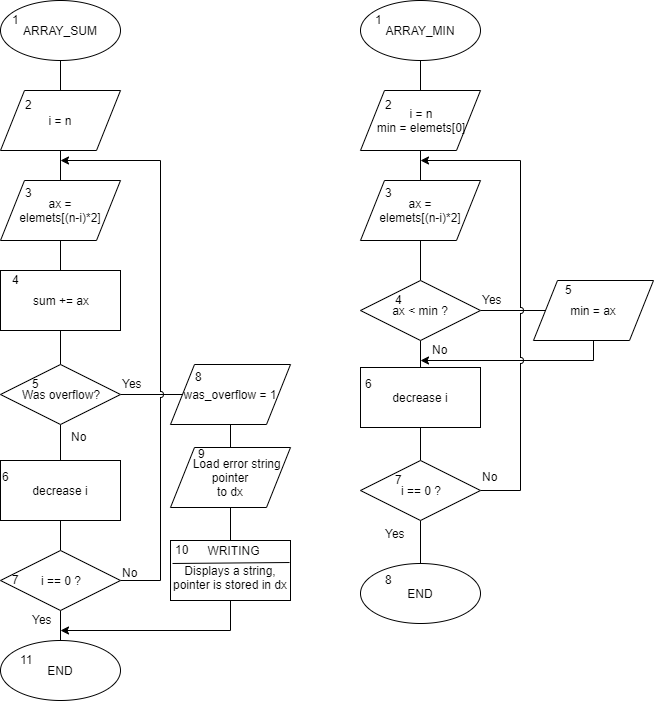
**E:\Developing\Assembler\Звіти\lab04\Screenshots\Invalid_length.PNGE:\Developing\Assembler\Звіти\lab04\Screenshots\Invalid_values.PNG**

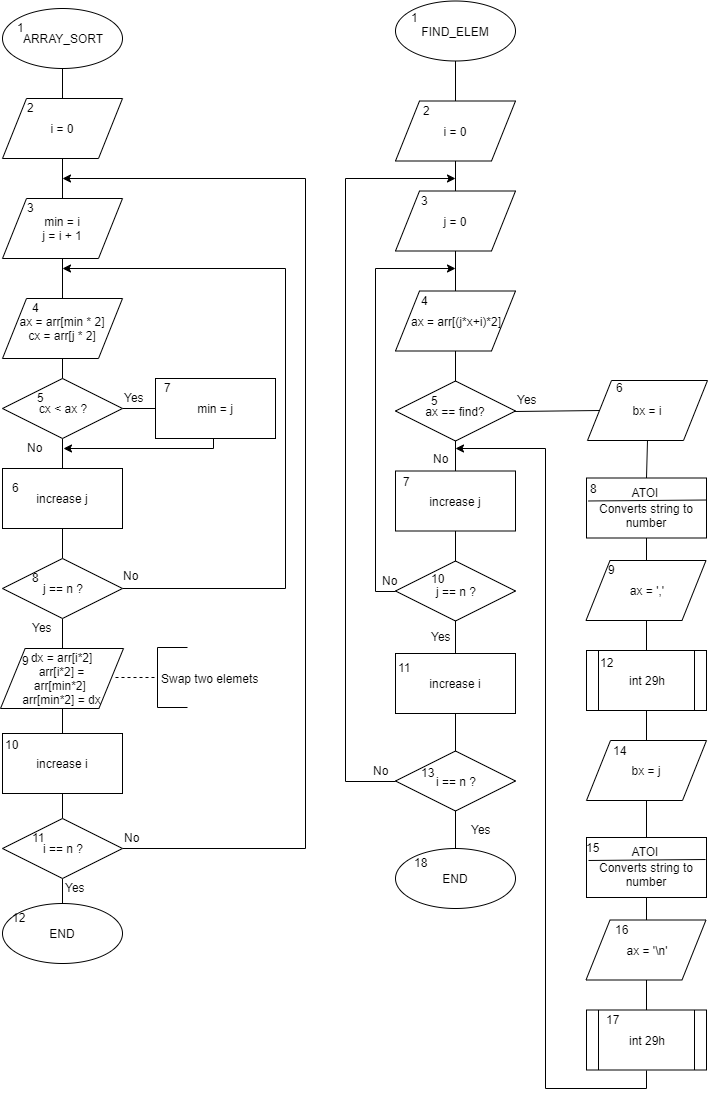
**Блок-схема**:

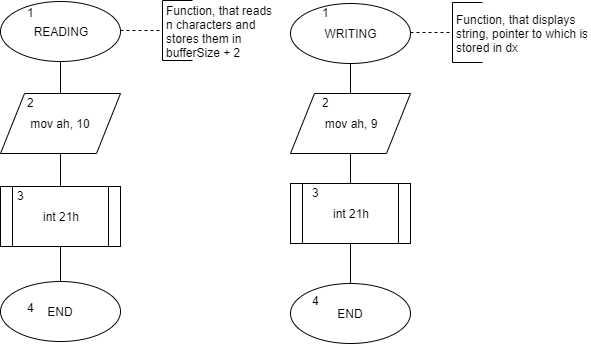












**Висновки**:

1. Були розібрані основні принципи роботи з масивами
2. Були вдосконалені навички написання програм з розгалуженнями
3. Був написаний структурований код з використанням окремих процедур, які можна використовувати в інших програмах
4. Були вдосконалені знання про синтаксис мови асемблер