

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з комп'ютерного практикуму № 5 з дисципліни

«Системне програмне забезпечення»

«Застосування макрозасобів мови асемблер»

Виконав:
студент групи ІП-11
Лесів В. І.

Перевірив:
Лісовиченко О. І.
«22» травня 2023 р.

Київ 2023

Комп'ютерний практикум 5

Застосування макрозасобів мови асемблер

Постановка завдання.

Скласти програми на нижче наведені завдання:

1. переписати програму комп'ютерного практикуму №2 з використанням одного макросу;
2. переписати програму комп'ютерного практикуму №3 з використанням макросів та передачею параметрів в них;
3. переписати одну програму (на вибір студента) комп'ютерного практикуму № 4 з використанням макросів та залученням міток в тілі макросу.

Хід роботи.**Текст програми 1.**

```
output_number macro

    mov bx, number

    or bx, bx

    jns m1

    mov al, '-'

    int 29h

    neg bx

m1:

    mov ax, bx

    xor cx, cx

    mov bx, 10

m2:

    xor dx, dx

    div bx

    add dl, '0'

    push dx

    inc cx

    test ax, ax

    jnz m2

m3:

    pop ax

    int 29h
```

loop m3

endm

STSEG SEGMENT PARA STACK "STACK"

DB 64 DUP("STACK")

STSEG ENDS

DSEG SEGMENT PARA PUBLIC "DATA"

enter_number_mes db 13, 10, "Enter number from [-4681; 4681]: \$"

invalid_input_mes db 13, 10, "Invalid input\$"

result_mes db 13, 10, "Result of multiplying by 7: \$"

input_number db 7, ?, 7 dup ('?')

number dw 0

is_negative db 0

is_error db 0

digit dw 0

DSEG ENDS

CSEG SEGMENT PARA PUBLIC "CODE"

ASSUME CS:CSEG, DS:DSEG, SS:STSEG

main proc

mov ax, dseg

mov ds, ax

```
lea dx, enter_number_mes
```

```
mov ah, 9
```

```
int 21h
```

```
call read
```

```
call transform
```

```
cmp is_error, 1
```

```
je error
```

```
mov ax, number
```

```
mov bx, 7
```

```
imul bx
```

```
mov number, ax
```

```
lea dx, result_mes
```

```
mov ah,9
```

```
int 21h
```

```
output_number
```

```
jmp end_program
```

```
error:
```

```
    lea dx, invalid_input_mes
```

```
    mov ah, 9
```

```
    int 21h
```

```
end_program:
```

```

        mov ah, 4CH

        int 21H

        ret

    ret

main endp

read proc

    lea dx, input_number

    mov ah, 10

    int 21h

; переходимо на наступний рядок після вводу

    mov al,10

    int 29h

    mov al,13

    int 29h

    ret

read endp

transform proc

    mov si, offset input_number + 2 ; завантажуюємо адресу input_number +
2, тобто перший елемент

    mov cx, 0

convert_loop:

; перевіряємо, чи кінець рядка

    mov ax, 0

    mov al, input_number + 1

```

cmp al, cl

je finish

mov al, [si] ; якщо ні, беремо символ цього рядка

cmp al, '0'

jl negative_sign

cmp al, '9'

jg error1

inc cx

inc si ; переходимо до наступного елемента

jmp transform_number

negative_sign:

cmp al, '-'

jne error1

; перевіряємо, чи - стоїть на початку рядка

cmp cx, 0

jne error1

mov is_negative, 1

inc cx

inc si

jmp convert_loop

transform_number:

sub al, '0'

mov digit, ax

mov bx, 10

```
    mov ax, number
    mul bx
    jc error1
    js error1
    mov number, ax
    mov ax, digit
    add number, ax
    jc error1
    js error1
    jmp convert_loop
error1:
    mov is_error, 1
    jmp end_prog

finish:
    cmp is_negative, 1
    jne end_prog
    neg number
end_prog:
    ret
transform endp
cseg ends
end main
```


Текст програми 2.

```
output_message macro message
```

```
    LEA dx, message
```

```
    MOV ah,9
```

```
    INT 21h
```

```
endm
```

```
read macro
```

```
    lea dx, input_number
```

```
    mov ah, 10
```

```
    int 21h
```

```
; переходимо на наступний рядок після вводу
```

```
    mov al,10
```

```
    int 29h
```

```
    mov al,13
```

```
    int 29h
```

```
endm
```

```
transform macro input_number
```

```
    local convert_loop, negative_sign, transform_number, error1, finish,  
end_prog
```

```
    mov si, offset input_number + 2 ; завантажуюмо адресу input_number +  
2, тобто перший елемент
```

```
mov cx, 0
```

```
convert_loop:
```

```
; перевіряємо, чи кінець рядка
```

```
    mov ax, 0
```

```
    mov al, input_number + 1
```

```
    cmp al, cl
```

```
    je finish
```

```
    mov al, [si] ; якщо ні, беремо символ цього рядка
```

```
    cmp al, '0'
```

```
    jl negative_sign
```

```
    cmp al, '9'
```

```
    jg error1
```

```
    inc cx
```

```
    inc si ; переходимо до наступного елемента
```

```
    jmp transform_number
```

```
negative_sign:
```

```
    cmp al, '-'
```

```
    jne error1
```

```
; перевіряємо, чи - стоїть на початку рядка
```

```
    cmp cx, 0
```

```
    jne error1
```

```
mov is_negative, 1
```

```
inc cx
```

```
inc si
```

```
jmp convert_loop
```

```
transform_number:
```

```
sub al, '0'
```

```
mov digit, ax
```

```
mov bx, 10
```

```
mov ax, number
```

```
mul bx
```

```
jc error1
```

```
js error1
```

```
mov number, ax
```

```
mov ax, digit
```

```
add number, ax
```

```
jc error1
```

```
js error1
```

```
jmp convert_loop
```

```
error1:
```

```
mov is_error, 1
```

```
jmp end_prog
```

finish:

 cmp is_negative, 1

 jne end_prog

 neg number

end_prog:

endm

mov_number macro numFrom, numTo

 mov ax, numFrom

 mov numTo, ax

 mov numFrom, 0

endm

calculate_x macro x

 local case1, case2, case3, error2, calculated

 cmp x, -1

 jle case3

 cmp x, 1

 jle case2

 cmp x, 3

 jle case1

 jmp calculated

case1:

; $z = 35/x + x^3 = 35/x + x * x * x$

mov ax, 35

div x

mov number, ax

mov remainder, dx

mov bx, x

mov ax, x

mul bx

jc error2

mov bx, x

mul bx

jc error2

add ax, number

mov number, ax

jmp calculated

case2:

; $z = x/(1 + x * x)$

mov bx, x

mov ax, x

mul bx

jc error2

```
add ax, 1  
  
mov number, ax  
  
mov ax, x  
  
div number  
  
mov remainder, dx  
  
mov number, ax  
  
jmp calculated
```

case3:

; z=2*x

```
mov bx, 2  
  
mov ax, x  
  
imul bx  
  
jo error2  
  
mov number, ax  
  
jmp calculated
```

error2:

```
mov is_error, 1
```

calculated:

endm

```

calculate_xy macro x,y

    local error21, calculated1

    mov ax,x

    add ax,y

    jo error21

    mov number, ax

    jmp calculated1

error21:

    mov is_error, 1

calculated1:

endm

```

```

output_result macro num

    local m1,m2,m3

    mov bx, num

    or bx, bx

    jns m1

    mov al, '-'

    int 29h

    neg bx

m1:

    mov ax, bx

    xor cx, cx

```

```
        mov bx, 10

m2:
        xor dx, dx

        div bx

        add dl, '0'

        push dx

        inc cx

        test ax, ax

        jnz m2

m3:
        pop ax

        int 29h

        loop m3

endm


STSEG SEGMENT PARA STACK "STACK"

        DB 64 DUP("STACK")

STSEG ENDS


DSEG SEGMENT PARA PUBLIC "DATA"

        x dw 0

        y dw 0

        input_number db 7,?,7 dup (" $")
```



```
welcome db "Enter numbers: $"

enter_x db 13, 10, "x = $"

enter_y db "y = $"

is_negative db 0

number dw 0

digit dw 0

is_error db 0

error_msg db "Error!$"

remainder dw 0

remainder_msg db " remainder $"

DSEG ENDS
```

```
CSEG SEGMENT PARA PUBLIC "CODE"
```

```
ASSUME CS:CSEG, DS:DSEG, SS:STSEG
```

```
main proc
```

```
.386
```

```
    MOV AX, DSEG
```

```
    MOV DS, AX
```

```
    output_message welcome
```

```
    output_message enter_x
```

```
    read
```

```
    transform input_number
```

```
    cmp is_error, 1
```

je error

; записуємо значення number до x і обнуляємо number

mov_number number, x

mov is_negative, 0

cmp x, 3

jg with_y

calculate_x x

jmp continuation

; вВОДИМО y

with_y:

output_message enter_y

read

transform input_number

cmp is_error, 1

je error

; записуємо number в y і обнуляємо

mov_number number, y

calculate_xy x,y

continuation:

cmp is_error, 1

je error

output_result number

cmp remainder, 0

je end_program

output_message remainder_msg

output_result remainder

jmp end_program

error:

output_message error_msg

end_program:

mov AH, 4CH

int 21H

ret

main endp

CSEG ENDS

end main

Текст програми 3. Для переписування я обрав частину КП №4 з двовимірним масивом.

```
output_message macro msg
```

```
    lea dx, msg
```

```
    mov ah, 9
```

```
    int 21h
```

```
endm
```

```
read macro
```

```
    lea dx, input_number
```

```
    mov ah, 10
```

```
    int 21h
```

```
endm
```

```
transform macro
```

```
    local convert_loop, negative_sign, transform_number, error1, finish,  
end_prog
```

```
    mov number, 0
```

```
    mov is_negative, 0
```

```
    mov si, offset input_number + 2 ; завантажуюємо адресу input_number +  
2, тобто перший елемент
```

```
    mov ch, 0
```

```
convert_loop:
```

```
    ; перевіряємо, чи кінець рядка
```

```
mov ax, 0

mov al, input_number + 1

cmp al, ch

je finish

mov al, [si] ; якщо ні, беремо символ цього рядка


cmp al, '0'

jl negative_sign

cmp al, '9'

jg error1

inc ch

inc si ; переходимо до наступного елемента

jmp transform_number
```

negative_sign:

```
cmp al, '-'

jne error1

; перевіряємо, чи - стоїть на початку рядка

cmp ch, 0

jne error1

mov is_negative, 1

inc ch

inc si
```

jmp convert_loop

transform_number:

sub al, '0'

mov digit, ax

mov bx, 10

mov ax, number

mul bx

jc error1

js error1

mov number, ax

mov ax, digit

add number, ax

jc error1

js error1

jmp convert_loop

error1:

mov is_error, 1

jmp end_prog

finish:

cmp is_negative, 1

jne end_prog

neg number

end_prog:

xor ch, ch

endm

mov_number macro numFrom, numTo

mov ax, numFrom

mov numTo, ax

endm

output_result macro num

local m1,m2,m3

mov bx, num

or bx, bx

jns m1

mov al, '-'

int 29h

neg bx

m1:

mov ax, bx

xor cx, cx

mov bx, 10

m2:

```
        xor dx, dx

        div bx

        add dl, '0'

        push dx

        inc cx

        test ax, ax

        jnz m2

m3:

        pop ax

        int 29h

        loop m3

endm


enter_matrix macro

    local enter_element, errorNum, end_input

    output_message input_msg

    mov di, 0

enter_element:

    output_message matrix_msg

    mov_number i, number

    add number, 1
```


output_result number

mov number, 0

output_message inner_brackets

mov_number j, number

add number, 1

output_result number

mov number, 0

output_message last_bracket

read

transform

cmp is_error, 1

je errorNum

mov_number number, [array+di]

;інкрементую покажчики

add di, 2

inc j

;якщо кінець рядка, переходимо до першого в наступному

mov cx, j

cmp cx, m

jl enter_element

mov j, 0

```

    inc i

    mov bx, i

    mov ax, 128

    mul bl

    mov di, ax


    cmp bx, n

    jl enter_element

    jmp end_input

```

```
errorNum:
```

```

    mov is_error, 0

    output_message error_msg

    jmp enter_element

```

```
end_input:
```

```
endm
```

```
find_element macro
```

```
    local dialog_loop, continue_dialog, not_found, end_proc
```

```
    dialog_loop:
```

```

        mov is_error, 0

        mov exists, 0

```

output_message find_value_msg

read

transform

cmp is_error, 1

je not_found

output_found_values

cmp exists, 0

je not_found

continue_dialog:

output_message retry_msg

read

mov dl, [input_number+2]

cmp dl, 'y'

je dialog_loop

jne end_proc

not_found:

output_message not_found_msg

```
    jmp continue_dialog
```

```
end_proc:
```

```
endm
```

```
output_found_values macro
```

```
    local find_entries, matr_loop, update_coord, found_output
```

```
    mov_number number, tmp
```

```
    find_entries:
```

```
        mov i, 0
```

```
        mov j, 0
```

```
        mov di, 0
```

```
    matr_loop:
```

```
        mov dx, [array+di]
```

```
        cmp dx, tmp
```

```
        je found_output
```

```
    update_coord:
```

```
        add di, 2
```

```
        inc j
```

```
        mov cx, j
```

```
    cmp cx, m
    jl matr_loop
    mov j, 0
    inc i
    mov bx, i
    mov ax, 128
    mul bl
    mov di, ax

    cmp bx, n
    jl matr_loop
    jmp the_end

found_output:
    mov exists, 1

    output_message matrix_msg

    mov_number i, number
    add number, 1
    output_result number
    output_message inner_brackets
    mov_number j, number
    add number, 1
```

```

        output_result number

        mov al, ']'

        int 29h

        jmp update_coord

    the_end:

endm

STSEG SEGMENT PARA STACK "STACK"

DB 64 DUP("STACK")

STSEG ENDS

DSEG SEGMENT PARA PUBLIC "DATA"

n dw 0 ; rows

m dw 0 ; columns

input_number db 7,?,7 dup (" $")

matrix_msg db 13, 10, "matr[$"

inner_brackets db "][$"

last_bracket db "]" => $"

is_negative db 0

number dw 0

digit dw 0

is_error db 0

error_msg db 13, 10, "Error.$"

n_msg db "Enter number of rows (1-64): $"

```

```

m_msg db 13, 10, "Enter number of columns (1-64): $"
input_msg db 13, 10, "Enter elements ([-32767; 32767])$"
array dw 64 dup (64 dup (?))

i dw 0

j dw 0

find_value_msg db 13, 10, "What value do you want to find? : $"
not_found_msg db 13, 10, "Element is not found. $"
retry_msg db 13, 10, "Continue the search? ('y' for yes): $"
exists db 0

tmp dw 0

DSEG ENDS

```

```

CSEG SEGMENT PARA PUBLIC "CODE"

```

```

ASSUME CS:CSEG, DS:DSEG, SS:STSEG

```

```

main proc

```

```

.386

```

```

    mov ax, dseg

```

```

    mov ds, ax

```

```

rest_n:

```

```

    mov is_error, 0

```

```

    output_message n_msg

```

```

read

```

```
transform

cmp is_error, 1

je restart_n

cmp number, 0

jle restart_n

cmp number, 64

jg restart_n

mov_number number, n


rest_m:

mov is_error, 0

output_message m_msg


read

transform

cmp is_error, 1

je restart_m

cmp number, 0

jle restart_m

cmp number, 64

jg restart_m

mov_number number, m


enter_matrix
```


find_element

jmp end_program

restart_n:

output_message error_msg

jmp rest_n

restart_m:

output_message error_msg

jmp rest_m

end_program:

mov AH, 4CH

int 21H

ret

main endp

CSEG ENDS

end main

Лістинг, зокрема міток у програмі 3. Створені унікальні мітки виділені жирним у лістингу.

Turbo Assembler Version 2.51

05/22/23 14:31:38

Page 1

lab5-3.ASM

```

1          output_message macro msg
2
3          lea dx, msg
4
5          mov ah, 9
6
7          int 21h
8
9          endm
10
11         read macro
12
13         lea dx, input_number
14
15         mov ah, 10
16
17         int 21h
18
19         endm
20
21         transform macro
22
23         local    convert_loop,    negative_sign,
24         transform_number, error1, finish, end_prog
25
26         mov number, 0
27
28         mov is_negative, 0
29
30         mov si, offset    input_number + 2 ;
31         завантажуюємо адресу    +
32
33         input_number + 2, тобто перший елемент

```

19	mov ch, 0	
20	convert_loop:	
21	; перевіряємо, чи кінець рядка	
22	mov ax, 0	
23	mov al, input_number + 1	
24	cmp al, ch	
25	je finish	
26	mov al, [si] ;	якщо ні,
беремо символ цього	+	
27	рядка	
28		
29	cmp al, '0'	
30	jl negative_sign	
31	cmp al, '9'	
32	jg error1	
33	inc ch	
34	inc si ;	переходимо до
наступного елемента		
35	jmp transform_number	
36		
37		
38	negative_sign:	
39	cmp al, '-'	
40	jne error1	

41		; перевіряємо, чи - стоїть на
початку рядка		
42		cmp ch, 0
43		jne error1
44		mov is_negative, 1
45		inc ch
46		inc si
47		jmp convert_loop
48		
49	transform_number:	
50		sub al, '0'
51		mov digit, ax
52		mov bx, 10
53		mov ax, number
54		mul bx
55		jc error1
56		js error1
57		mov number, ax

lab5-3.ASM

```
58             mov ax, digit
59             add number, ax
60             jc error1
61             js error1
62             jmp convert_loop
63
64             error1:
65             mov is_error, 1
66             jmp end_prog
67
68             finish:
69             cmp is_negative, 1
70             jne end_prog
71             neg number
72
73             end_prog:
74             xor ch, ch
75             endm
76
```

```
77         mov_number macro numFrom, numTo
78             mov ax, numFrom
79             mov numTo, ax
80         endm
81
82         output_result macro num
83             local m1,m2,m3
84             mov bx, num
85             or bx,      bx
86             jns m1
87             mov al, '-'
88             int 29h
89             neg bx
90             m1:
91                 mov ax, bx
92                 xor cx, cx
93                 mov bx, 10
94             m2:
95                 xor dx, dx
96                 div bx
97                 add dl, '0'
98                 push dx
99                 inc cx
100                test ax, ax
```

```
101                                jnz m2
102                                m3:
103                                pop ax
104                                int 29h
105                                loop m3
106                                endm
107
108
109                                enter_matrix macro
110                                local    enter_element,    errorNum,
end_input
111                                output_message    input_msg
112
113                                mov di, 0
114
```

lab5-3.ASM

```
115          enter_element:
116              output_message  matrix_msg
117              mov_number i, number
118              add number, 1
119              output_result number
120              mov number, 0
121              output_message  inner_brackets
122              mov_number j, number
123              add number, 1
124              output_result number
125              mov number, 0
126              output_message  last_bracket
127
128              read
129              transform
130              cmp is_error, 1
131              je errorNum
132
133              mov_number number, [array+di]
```


134	
135	;інкрементую покажчики
136	add di, 2
137	inc j
138	;якщо кінець рядка, переходимо до
першого в +	
139	наступному
140	mov cx, j
141	cmp cx, m
142	jl enter_element
143	mov j, 0
144	inc i
145	mov bx, i
146	mov ax, 128
147	mul bl
148	mov di, ax
149	
150	cmp bx, n
151	jl enter_element
152	jmp end_input
153	
154	errorNum:
155	mov is_error, 0
156	output_message error_msg

```
157                                     jmp enter_element
158
159                                     end_input:
160                                endm
161
162                                find_element macro
163                                local    dialog_loop,    continue_dialog,
not_found, end_proc
164                                dialog_loop:
165                                mov is_error, 0
166                                mov exists, 0
167
168                                output_message    find_value_msg
169
170                                read
171                                transform
```

lab5-3.ASM

```
172
173             cmp is_error, 1
174             je not_found
175
176             output_found_values
177             cmp exists, 0
178             je not_found
179
180             continue_dialog:
181             output_message  retry_msg
182
183             read
184
185             mov dl, [input_number+2]
186             cmp dl, 'y'
187             je dialog_loop
188             jne end_proc
189
190             not_found:
191             output_message  not_found_msg
192             jmp continue_dialog
193
```

```
194                     end_proc:
195                     endm
196
197
198                     output_found_values macro
199                         local      find_entries,      matr_loop,
update_coord, found_output
200                         mov_number number, tmp
201
202                         find_entries:
203                             mov i,      0
204                             mov j,      0
205                             mov di, 0
206
207                         matr_loop:
208                             mov dx, [array+di]
209                             cmp dx, tmp
210                             je found_output
211
212                         update_coord:
213                             add di, 2
214                             inc j
215                             mov cx, j
216                             cmp cx, m
```

```
217                jl matr_loop
218                mov j,      0
219                inc i
220                mov bx, i
221                mov ax, 128
222                mul bl
223                mov di, ax
224
225                cmp bx, n
226                jl matr_loop
227                jmp the_end
228
```

lab5-3.ASM

```
229             found_output:
230                 mov exists, 1
231
232                 output_message  matrix_msg
233
234                 mov_number i, number
235                 add number, 1
236                 output_result number
237                 output_message  inner_brackets
238                 mov_number j, number
239                 add number, 1
240                 output_result number
241                 mov al, ']'
242                 int 29h
243                 jmp update_coord
244             the_end:
245         endm
246
247 0000             STSEG SEGMENT PARA STACK "STACK"
248 0000 40*(53 54 41 43 4B)  DB 64 DUP("STACK")
249 0140             STSEG ENDS
250
```

```

251 0000                                DSEG SEGMENT PARA PUBLIC "DATA"

252 0000 0000                                n dw 0      ; rows

253 0002 0000                                m dw 0      ; columns

254 0004 07 ??      07*(20 24)  input_number db 7,?,7 dup (" $")

255 0014 0D 0A      6D 61 74 72 5B+  matrix_msg db 13, 10, "matr[$"

256      24

257 001C 5D 5B      24      inner_brackets  db ")][$"

258 001F 5D 20      3D 3E 20 24      last_bracket db "]" => $"

259 0025 00      is_negative db 0

260 0026 0000      number      dw 0

261 0028 0000      digit dw 0

262 002A 00      is_error db 0

263 002B 0D 0A      45 72 72 6F 72+  error_msg db 13, 10, "Error.$"

264      2E 24

265 0034 45 6E      74 65 72 20 6E+  n_msg db "Enter number      of
rows (1-64):      $"

266      75 6D      62 65 72 20 6F+

267      66 20      72 6F 77 73 20+

268      28 31      2D 36 34 29 3A+

269      20 20      24

270 0053 0D 0A      45 6E 74 65 72+  m_msg db 13, 10, "Enter number
of columns (1-64): $"

271      20 6E      75 6D 62 65 72+

272      20 6F      66 20 63 6F 6C+

```

```

273      75 6D      6E 73 20 28 31+
274      2D 36      34 29 3A 20 24
275  0076 0D 0A    45 6E 74 65 72+  input_msg  db  13,  10,  "Enter
elements ([-32767;32767])$"
276      20 65      6C 65 6D 65 6E+
277      74 73      20 28 5B 2D 33+
278      32 37      36 37 3B 20 33+
279      32 37      36 37 5D 29 24
280  0099 40*(40*(????))  array dw 64 dup (64 dup (?))
281  2099 0000                                i dw 0
282  209B 0000                                j dw 0
283  209D 0D 0A    57 68 61 74 20+  find_value_msg  db  13,  10,
"What value do you want to find? : $"
284      76 61      6C 75 65 20 64+
285      6F 20      79 6F 75 20 77+

```


lab5-3.ASM

```

286      61 6E      74 20 74 6F 20+
287      66 69      6E 64 3F 20 3A+
288      20 24
289  20C2 0D 0A  45 6C 65 6D 65+  not_found_msg  db  13,  10,
"Element is not found. $"
290      6E 74      20 69 73 20 6E+
291      6F 74      20 66 6F 75 6E+
292      64 2E      20 24
293  20DB 0D 0A  43 6F 6E 74 69+  retry_msg db 13, 10, "Continue
the search? ('y' for yes): $"
294      6E 75      65 20 74 68 65+
295      20 73      65 61 72 63 68+
296      3F 20      28 27 79 27 20+
297      66 6F      72 20 79 65 73+
298      29 3A      20 24
299  2102 00                      existsdb 0
300  2103 0000                      tmp dw    0
301  2105                      DSEG ENDS
302
303  0000                      CSEG SEGMENT PARA PUBLIC "CODE"
304                      ASSUME  CS:CSEG, DS:DSEG, SS:STSEG
305  0000                      main proc

```

```

306                                     .386

307  0000 B8 0000s                      mov ax, dseg

308  0003 8E D8                          mov ds, ax

309

310  0005                                rest_n:

311  0005 C6 06    002Ar 00              mov is_error, 0

312                                output_message  n_msg

1 313      000A BA 0034r                  lea dx, n_msg

1 314      000D B4 09                      mov ah, 9

1 315      000F CD 21                      int 21h

316

317                                read

1 318      0011 BA 0004r                  lea dx, input_number

1 319      0014 B4 0A                      mov ah, 10

1 320      0016 CD 21                      int 21h

321                                transform

1 322      0018 C7 06    0026r 0000      mov number, 0

1 323      001E C6 06    0025r 00      mov is_negative, 0

1 324      0023 BE 0006r                  mov si, offset    input_number +
2 ; завантажуюмо адресу          +

325                                input_number + 2, тобто перший елемент

1 326      0026 B5 00                      mov ch, 0

1 327      0028                                ??0000:

1 328      0028 B8 0000                      mov ax, 0

```

1	329	002B A0 0005r	mov al, input_number + 1
1	330	002E 3A C5	cmp al, ch
1	331	0030 74 5E 90 90	je ??0004
1	332	0034 8A 04	mov al, [si] ;
якщо ні, беремо символ цього +			
	333	рядка	
1	334	0036 3C 30	cmp al, '0'
1	335	0038 7C 0E 90 90	jl ??0001
1	336	003C 3C 39	cmp al, '9'
1	337	003E 7F 48 90 90	jg ??0003
1	338	0042 FE C5	inc ch
1	339	0044 46	inc si; переходимо до
наступного елемента			
1	340	0045 EB 18 90	jmp ??0002
1	341	0048	??0001:
1	342	0048 3C 2D	cmp al, '-'

lab5-3.ASM

```

1 343      004A 75 3C   90 90                jne ??0003
1 344      004E 80 FD   00                  cmp ch, 0
1 345      0051 75 35   90 90                jne ??0003
1 346      0055 C6 06   0025r 01          mov is_negative, 1
1 347      005A FE C5                      inc ch
1 348      005C 46                      inc si
1 349      005D EB C9                      jmp ??0000
1 350      005F                                ??0002:
1 351      005F 2C 30                      sub al, '0'
1 352      0061 A3 0028r                  mov digit, ax
1 353      0064 BB 000A                  mov bx, 10
1 354      0067 A1 0026r                  mov ax, number
1 355      006A F7 E3                      mul bx
1 356      006C 72 1A   90 90                jc ??0003
1 357      0070 78 16   90 90                js ??0003
1 358      0074 A3 0026r                  mov number, ax
1 359      0077 A1 0028r                  mov ax, digit
1 360      007A 01 06   0026r                  add number, ax
1 361      007E 72 08   90 90                jc ??0003
1 362      0082 78 04   90 90                js ??0003
1 363      0086 EB A0                      jmp ??0000
1 364      0088                                ??0003:

```

```

1 365      0088 C6 06    002Ar 01      mov is_error, 1
1 366      008D EB 0E    90              jmp ??0005
1 367      0090              ??0004:
1 368      0090 80 3E    0025r 01      cmp is_negative, 1
1 369      0095 75 06    90 90      jne ??0005
1 370      0099 F7 1E    0026r      neg number
1 371      009D              ??0005:
1 372      009D 32 ED              xor ch, ch

373 009F 80 3E    002Ar 01      cmp is_error, 1
374 00A4 0F 84    03F2      je restart_n
375 00A8 83 3E    0026r 00      cmp number, 0
376 00AD 0F 8E    03E9      jle restart_n
377 00B1 83 3E    0026r 40      cmp number, 64
378 00B6 0F 8F    03E0      jg restart_n
379              mov_number number, n

1 380      00BA A1 0026r      mov ax, number
1 381      00BD A3 0000r      mov n,      ax
382
383 00C0              rest_m:
384 00C0 C6 06    002Ar 00      mov is_error, 0
385              output_message m_msg
1 386      00C5 BA 0053r      lea dx, m_msg
1 387      00C8 B4 09              mov ah, 9
1 388      00CA CD 21              int 21h

```

```

389
390                                     read
1 391      00CC BA 0004r                lea dx, input_number
1 392      00CF B4 0A                mov ah, 10
1 393      00D1 CD 21                int 21h
394                                     transform
1 395      00D3 C7 06 0026r 0000      mov number, 0
1 396      00D9 C6 06 0025r 00      mov is_negative, 0
1 397      00DE BE 0006r                mov      si,      offset
input_number + 2 ; завантажуюмо адресу      +
398                                     input_number + 2, тобто перший елемент
1 399      00E1 B5 00                mov ch, 0

```

lab5-3.ASM

```

1 400      00E3      ??0006:

1 401      00E3 B8 0000      mov ax, 0

1 402      00E6 A0 0005r      mov al, input_number + 1

1 403      00E9 3A C5      cmp al, ch

1 404      00EB 74 5E    90 90      je ??000A

1 405      00EF 8A 04      mov  al,  [si]  ;
якщо ні, беремо символ цього      +
406      рядка

1 407      00F1 3C 30      cmp al, '0'

1 408      00F3 7C 0E    90 90      jl ??0007

1 409      00F7 3C 39      cmp al, '9'

1 410      00F9 7F 48    90 90      jg ??0009

1 411      00FD FE C5      inc ch

1 412      00FF 46      inc si;   переходимо   до
наступного елемента

1 413      0100 EB 18    90      jmp ??0008

1 414      0103      ??0007:

1 415      0103 3C 2D      cmp al, '-'

1 416      0105 75 3C    90 90      jne ??0009

1 417      0109 80 FD    00      cmp ch, 0

1 418      010C 75 35    90 90      jne ??0009

1 419      0110 C6 06    0025r 01      mov is_negative, 1

```

1	420	0115 FE C5		inc ch
1	421	0117 46		inc si
1	422	0118 EB C9		jmp ??0006
1	423	011A		??0008:
1	424	011A 2C 30		sub al, '0'
1	425	011C A3 0028r		mov digit, ax
1	426	011F BB 000A		mov bx, 10
1	427	0122 A1 0026r		mov ax, number
1	428	0125 F7 E3		mul bx
1	429	0127 72 1A	90 90	jc ??0009
1	430	012B 78 16	90 90	js ??0009
1	431	012F A3 0026r		mov number, ax
1	432	0132 A1 0028r		mov ax, digit
1	433	0135 01 06	0026r	add number, ax
1	434	0139 72 08	90 90	jc ??0009
1	435	013D 78 04	90 90	js ??0009
1	436	0141 EB A0		jmp ??0006
1	437	0143		??0009:
1	438	0143 C6 06	002Ar 01	mov is_error, 1
1	439	0148 EB 0E	90	jmp ??000B
1	440	014B		??000A:
1	441	014B 80 3E	0025r 01	cmp is_negative, 1
1	442	0150 75 06	90 90	jne ??000B
1	443	0154 F7 1E	0026r	neg number


```

1 444          0158          ??000B:

1 445          0158 32 ED                      xor ch, ch

446 015A 80 3E    002Ar 01          cmp is_error, 1

447 015F 0F 84    0341          je restart_m

448 0163 83 3E    0026r 00          cmp number, 0

449 0168 0F 8E    0338          jle restart_m

450 016C 83 3E    0026r 40          cmp number, 64

451 0171 0F 8F    032F          jg restart_m

452                                mov_number number, m

1 453          0175 A1 0026r          mov ax, number

1 454          0178 A3 0002r          mov m,    ax

455

456                                enter_matrix

```

lab5-3.ASM

```

1 457                                     output_message input_msg
2 458      017B BA 0076r                  lea dx, input_msg
2 459      017E B4 09                      mov ah, 9
2 460      0180 CD 21                      int 21h
1 461      0182 BF 0000                  mov di, 0
1 462      0185                          ??000C:
1 463                                     output_message
matrix_msg
2 464      0185 BA 0014r                  lea dx, matrix_msg
2 465      0188 B4 09                      mov ah, 9
2 466      018A CD 21                      int 21h
1 467                                     mov_number i, number
2 468      018C A1 2099r                  mov ax, i
2 469      018F A3 0026r                  mov number, ax
1 470      0192 83 06    0026r 01          add number, 1
1 471                                     output_result number
2 472      0197 8B 1E    0026r            mov bx, number
2 473      019B 0B DB                      or bx,    bx
2 474      019D 79 08    90 90            jns ??000F

```

2	475	01A1 B0 2D	mov al, '-'
2	476	01A3 CD 29	int 29h
2	477	01A5 F7 DB	neg bx
2	478	01A7	??000F:
2	479	01A7 8B C3	mov ax, bx
2	480	01A9 33 C9	xor cx, cx
2	481	01AB BB 000A	mov bx, 10
2	482	01AE	??0010:
2	483	01AE 33 D2	xor dx, dx
2	484	01B0 F7 F3	div bx
2	485	01B2 80 C2 30	add dl, '0'
2	486	01B5 52	push dx
2	487	01B6 41	inc cx
2	488	01B7 85 C0	test ax, ax
2	489	01B9 75 F3	jnz ??0010
2	490	01BB	??0011:
2	491	01BB 58	pop ax
2	492	01BC CD 29	int 29h
2	493	01BE E2 FB	loop ??0011
1	494	01C0 C7 06 0026r 0000	mov number, 0
1	495		output_message
		inner_brackets	
2	496	01C6 BA 001Cr	lea dx, inner_brackets
2	497	01C9 B4 09	mov ah, 9

2	498	01CB CD 21	int 21h
1	499		mov_number j, number
2	500	01CD A1 209Br	mov ax, j
2	501	01D0 A3 0026r	mov number, ax
1	502	01D3 83 06 0026r 01	add number, 1
1	503		output_result number
2	504	01D8 8B 1E 0026r	mov bx, number
2	505	01DC 0B DB	or bx, bx
2	506	01DE 79 08 90 90	jns ??0012
2	507	01E2 B0 2D	mov al, '-'
2	508	01E4 CD 29	int 29h
2	509	01E6 F7 DB	neg bx
2	510	01E8	??0012:
2	511	01E8 8B C3	mov ax, bx
2	512	01EA 33 C9	xor cx, cx
2	513	01EC BB 000A	mov bx, 10

lab5-3.ASM

```

2  514      01EF      ??0013:

2  515      01EF 33 D2      xor dx, dx

2  516      01F1 F7 F3      div bx

2  517      01F3 80 C2    30      add dl, '0'

2  518      01F6 52      push dx

2  519      01F7 41      inc cx

2  520      01F8 85 C0      test ax, ax

2  521      01FA 75 F3      jnz ??0013

2  522      01FC      ??0014:

2  523      01FC 58      pop ax

2  524      01FD CD 29      int 29h

2  525      01FF E2 FB      loop ??0014

1  526      0201 C7 06    0026r 0000      mov number, 0

1  527      output_message

last_bracket

2  528      0207 BA 001Fr      lea dx, last_bracket

2  529      020A B4 09      mov ah, 9

2  530      020C CD 21      int 21h

1  531      read

2  532      020E BA 0004r      lea dx, input_number

2  533      0211 B4 0A      mov ah, 10

2  534      0213 CD 21      int 21h

```

1	535			transform
2	536	0215 C7 06	0026r 0000	mov number, 0
2	537	021B C6 06	0025r 00	mov is_negative, 0
2	538	0220 BE 0006r		mov si, offset input_number +
2 ; завантажуюмо адресу +				
	539			input_number + 2, тобто перший елемент
2	540	0223 B5 00		mov ch, 0
2	541	0225		??0015:
2	542	0225 B8 0000		mov ax, 0
2	543	0228 A0 0005r		mov al, input_number + 1
2	544	022B 3A C5		cmp al, ch
2	545	022D 74 5E	90 90	je ??0019
2	546	0231 8A 04		mov al, [si] ;
якщо ні, беремо символ цього +				
	547			рядка
2	548	0233 3C 30		cmp al, '0'
2	549	0235 7C 0E	90 90	jl ??0016
2	550	0239 3C 39		cmp al, '9'
2	551	023B 7F 48	90 90	jg ??0018
2	552	023F FE C5		inc ch
2	553	0241 46		inc si; переходимо до
наступного елемента				
2	554	0242 EB 18	90	jmp ??0017
2	555	0245		??0016:

2	556	0245 3C 2D	cmp al, '-'
2	557	0247 75 3C 90 90	jne ??0018
2	558	024B 80 FD 00	cmp ch, 0
2	559	024E 75 35 90 90	jne ??0018
2	560	0252 C6 06 0025r 01	mov is_negative, 1
2	561	0257 FE C5	inc ch
2	562	0259 46	inc si
2	563	025A EB C9	jmp ??0015
2	564	025C	??0017:
2	565	025C 2C 30	sub al, '0'
2	566	025E A3 0028r	mov digit, ax
2	567	0261 BB 000A	mov bx, 10
2	568	0264 A1 0026r	mov ax, number
2	569	0267 F7 E3	mul bx
2	570	0269 72 1A 90 90	jc ??0018

lab5-3.ASM

```

2 571      026D 78 16    90 90                js ??0018
2 572      0271 A3 0026r                mov number, ax
2 573      0274 A1 0028r                mov ax, digit
2 574      0277 01 06    0026r            add number, ax
2 575      027B 72 08    90 90                jc ??0018
2 576      027F 78 04    90 90                js ??0018
2 577      0283 EB A0                    jmp ??0015
2 578      0285                    ??0018:
2 579      0285 C6 06    002Ar 01          mov is_error, 1
2 580      028A EB 0E    90                jmp ??001A
2 581      028D                    ??0019:
2 582      028D 80 3E    0025r 01          cmp is_negative, 1
2 583      0292 75 06    90 90                jne ??001A
2 584      0296 F7 1E    0026r            neg number
2 585      029A                    ??001A:
2 586      029A 32 ED                    xor ch, ch
1 587      029C 80 3E    002Ar 01          cmp is_error, 1
1 588      02A1 74 3C    90 90                je ??000D

```


1	589			mov_number	number,
		[array+di]			
2	590	02A5 A1 0026r		mov ax, number	
2	591	02A8 89 85 0099r		mov [array+di], ax	
1	592	02AC 83 C7 02		add di, 2	
1	593	02AF FF 06 209Br		inc j	
1	594	02B3 8B 0E 209Br		mov cx, j	
1	595	02B7 3B 0E 0002r		cmp cx, m	
1	596	02BB 0F 8C FEC6		jl ??000C	
1	597	02BF C7 06 209Br 0000		mov j, 0	
1	598	02C5 FF 06 2099r		inc i	
1	599	02C9 8B 1E 2099r		mov bx, i	
1	600	02CD B8 0080		mov ax, 128	
1	601	02D0 F6 E3		mul bl	
1	602	02D2 8B F8		mov di, ax	
1	603	02D4 3B 1E 0000r		cmp bx, n	
1	604	02D8 0F 8C FEA9		jl ??000C	
1	605	02DC EB 10 90		jmp ??000E	
1	606	02DF		??000D:	
1	607	02DF C6 06 002Ar 00		mov is_error, 0	
1	608			output_message	
		error_msg			
2	609	02E4 BA 002Br		lea dx, error_msg	
2	610	02E7 B4 09		mov ah, 9	

2	611	02E9 CD 21	int 21h
1	612	02EB E9 FE97	jmp ??000C
1	613	02EE	??000E:
	614		
	615		find_element
1	616	02EE	??001B:
1	617	02EE C6 06 002Ar 00	mov is_error, 0
1	618	02F3 C6 06 2102r 00	mov exists, 0
1	619		output_message
			find_value_msg
2	620	02F8 BA 209Dr	lea dx, find_value_msg
2	621	02FB B4 09	mov ah, 9
2	622	02FD CD 21	int 21h
1	623		read
2	624	02FF BA 0004r	lea dx, input_number
2	625	0302 B4 0A	mov ah, 10
2	626	0304 CD 21	int 21h
1	627		transform

lab5-3.ASM

```

2 628      0306 C7 06      0026r 0000      mov number, 0

2 629      030C C6 06      0025r 00      mov is_negative, 0

2 630      0311 BE 0006r      mov si, offset      input_number +
2 ; завантажуюмо адресу      +

631      input_number + 2, тобто перший елемент

2 632      0314 B5 00      mov ch, 0

2 633      0316      ??001F:

2 634      0316 B8 0000      mov ax, 0

2 635      0319 A0 0005r      mov al, input_number + 1

2 636      031C 3A C5      cmp al, ch

2 637      031E 74 5E      90 90      je ??0023

2 638      0322 8A 04      mov al, [si] ;
якщо ні, беремо символ цього      +
639      рядка

2 640      0324 3C 30      cmp al, '0'

2 641      0326 7C 0E      90 90      jl ??0020

2 642      032A 3C 39      cmp al, '9'

2 643      032C 7F 48      90 90      jg ??0022

2 644      0330 FE C5      inc ch

2 645      0332 46      inc si;      переходимо до
наступного елемента

2 646      0333 EB 18      90      jmp ??0021

```

2	647	0336	??0020:	
2	648	0336 3C 2D		cmp al, '-'
2	649	0338 75 3C 90 90		jne ??0022
2	650	033C 80 FD 00		cmp ch, 0
2	651	033F 75 35 90 90		jne ??0022
2	652	0343 C6 06 0025r 01		mov is_negative, 1
2	653	0348 FE C5		inc ch
2	654	034A 46		inc si
2	655	034B EB C9		jmp ??001F
2	656	034D	??0021:	
2	657	034D 2C 30		sub al, '0'
2	658	034F A3 0028r		mov digit, ax
2	659	0352 BB 000A		mov bx, 10
2	660	0355 A1 0026r		mov ax, number
2	661	0358 F7 E3		mul bx
2	662	035A 72 1A 90 90		jc ??0022
2	663	035E 78 16 90 90		js ??0022
2	664	0362 A3 0026r		mov number, ax
2	665	0365 A1 0028r		mov ax, digit
2	666	0368 01 06 0026r		add number, ax
2	667	036C 72 08 90 90		jc ??0022
2	668	0370 78 04 90 90		js ??0022
2	669	0374 EB A0		jmp ??001F
2	670	0376	??0022:	

2	671	0376 C6 06	002Ar 01	mov is_error, 1
2	672	037B EB 0E	90	jmp ??0024
2	673	037E		??0023:
2	674	037E 80 3E	0025r 01	cmp is_negative, 1
2	675	0383 75 06	90 90	jne ??0024
2	676	0387 F7 1E	0026r	neg number
2	677	038B		??0024:
2	678	038B 32 ED		xor ch, ch
1	679	038D 80 3E	002Ar 01	cmp is_error, 1
1	680	0392 0F 84	00F8	je ??001D
1	681			output_found_values
2	682			mov_number number, tmp
3	683	0396 A1 0026r		mov ax, number
3	684	0399 A3 2103r		mov tmp, ax

lab5-3.ASM

```

2 685      039C      ??0025:

2 686      039C C7 06  2099r 0000      mov i,      0

2 687      03A2 C7 06  209Br 0000      mov j,      0

2 688      03A8 BF 0000      mov di, 0

2 689      03AB      ??0026:

2 690      03AB 8B 95  0099r      mov dx, [array+di]

2 691      03AF 3B 16  2103r      cmp dx, tmp

2 692      03B3 74 31  90 90      je ??0028

2 693      03B7      ??0027:

2 694      03B7 83 C7  02      add di, 2

2 695      03BA FF 06  209Br      inc j

2 696      03BE 8B 0E  209Br      mov cx, j

2 697      03C2 3B 0E  0002r      cmp cx, m

2 698      03C6 7C E3      jl ??0026

2 699      03C8 C7 06  209Br 0000      mov j,      0

2 700      03CE FF 06  2099r      inc i

2 701      03D2 8B 1E  2099r      mov bx, i

2 702      03D6 B8 0080      mov ax, 128

2 703      03D9 F6 E3      mul bl

2 704      03DB 8B F8      mov di, ax

2 705      03DD 3B 1E  0000r      cmp bx, n

2 706      03E1 7C C8      jl ??0026

```

2	707	03E3 E9 0082	jmp the_end
2	708	03E6	??0028:
2	709	03E6 C6 06 2102r 01	mov exists, 1
2	710		output_message
		matrix_msg	
3	711	03EB BA 0014r	lea dx, matrix_msg
3	712	03EE B4 09	mov ah, 9
3	713	03F0 CD 21	int 21h
2	714		mov_number i, number
3	715	03F2 A1 2099r	mov ax, i
3	716	03F5 A3 0026r	mov number, ax
2	717	03F8 83 06 0026r 01	add number, 1
2	718		output_result number
3	719	03FD 8B 1E 0026r	mov bx, number
3	720	0401 0B DB	or bx, bx
3	721	0403 79 08 90 90	jns ??0029
3	722	0407 B0 2D	mov al, '-'
3	723	0409 CD 29	int 29h
3	724	040B F7 DB	neg bx
3	725	040D	??0029:
3	726	040D 8B C3	mov ax, bx
3	727	040F 33 C9	xor cx, cx
3	728	0411 BB 000A	mov bx, 10
3	729	0414	??002A:

3	730	0414 33 D2	xor dx, dx
3	731	0416 F7 F3	div bx
3	732	0418 80 C2 30	add dl, '0'
3	733	041B 52	push dx
3	734	041C 41	inc cx
3	735	041D 85 C0	test ax, ax
3	736	041F 75 F3	jnz ??002A
3	737	0421	??002B:
3	738	0421 58	pop ax
3	739	0422 CD 29	int 29h
3	740	0424 E2 FB	loop ??002B
2	741		output_message

inner_brackets

lab5-3.ASM

```

3 742      0426 BA 001Cr      lea dx, inner_brackets
3 743      0429 B4 09      mov ah, 9
3 744      042B CD 21      int 21h
2 745      mov_number j, number
3 746      042D A1 209Br      mov ax, j
3 747      0430 A3 0026r      mov number, ax
2 748      0433 83 06      0026r 01      add number, 1
2 749      output_result number
3 750      0438 8B 1E      0026r      mov bx, number
3 751      043C 0B DB      or bx,      bx
3 752      043E 79 08      90 90      jns ??002C
3 753      0442 B0 2D      mov al, '-'
3 754      0444 CD 29      int 29h
3 755      0446 F7 DB      neg bx
3 756      0448      ??002C:
3 757      0448 8B C3      mov ax, bx
3 758      044A 33 C9      xor cx, cx
3 759      044C BB 000A      mov bx, 10
3 760      044F      ??002D:
3 761      044F 33 D2      xor dx, dx
3 762      0451 F7 F3      div bx
3 763      0453 80 C2      30      add dl, '0'

```

3	764	0456 52		push dx
3	765	0457 41		inc cx
3	766	0458 85 C0		test ax, ax
3	767	045A 75 F3		jnz ??002D
3	768	045C		??002E:
3	769	045C 58		pop ax
3	770	045D CD 29		int 29h
3	771	045F E2 FB		loop ??002E
2	772	0461 B0 5D		mov al, 'J'
2	773	0463 CD 29		int 29h
2	774	0465 E9 FF4F		jmp ??0027
2	775	0468		the_end:
1	776	0468 80 3E	2102r 00	cmp exists, 0
1	777	046D 74 1F	90 90	je ??001D
1	778	0471		??001C:
1	779			output_message retry_msg
2	780	0471 BA 20DBr		lea dx, retry_msg
2	781	0474 B4 09		mov ah, 9
2	782	0476 CD 21		int 21h
1	783			read
2	784	0478 BA 0004r		lea dx, input_number
2	785	047B B4 0A		mov ah, 10
2	786	047D CD 21		int 21h

```

1 787      047F 8A 16   0006r      mov      dl,
[input_number+2]

1 788      0483 80 FA   79      cmp dl, 'y'

1 789      0486 0F 84   FE64      je ??001B

1 790      048A 75 0B   90 90      jne ??001E

1 791      048E      ??001D:

1 792      output_message
not_found_msg

2 793      048E BA 20C2r      lea dx, not_found_msg

2 794      0491 B4 09      mov ah, 9

2 795      0493 CD 21      int 21h

1 796      0495 EB DA      jmp ??001C

1 797      0497      ??001E:

798 0497 EB 15   90      jmp end_program

```

lab5-3.ASM

```
799
800 049A          restart_n:
801                output_message error_msg
1 802          049A BA 002Br          lea dx, error_msg
1 803          049D B4 09          mov ah, 9
1 804          049F CD 21          int 21h
805 04A1 E9 FB61          jmp rest_n
806
807 04A4          restart_m:
808                output_message error_msg
1 809          04A4 BA 002Br          lea dx, error_msg
1 810          04A7 B4 09          mov ah, 9
1 811          04A9 CD 21          int 21h
812 04AB E9 FC12          jmp rest_m
813
814 04AE          end_program:
815 04AE B4 4C          mov AH, 4CH
816 04B0 CD 21          int 21H
817 04B2 C3          ret
```

818	04B3	main endp
819	04B3	CSEG ENDS
820		end main

Symbol Table

Symbol Name	Type	Value
??0000	Near CSEG:	0028
??0001	Near CSEG:	0048
??0002	Near CSEG:	005F
??0003	Near CSEG:	0088
??0004	Near CSEG:	0090
??0005	Near CSEG:	009D
??0006	Near CSEG:	00E3
??0007	Near CSEG:	0103
??0008	Near CSEG:	011A
??0009	Near CSEG:	0143
??000A	Near CSEG:	014B
??000B	Near CSEG:	0158
??000C	Near CSEG:	0185
??000D	Near CSEG:	02DF
??000E	Near CSEG:	02EE
??000F	Near CSEG:	01A7
??0010	Near CSEG:	01AE
??0011	Near CSEG:	01BB
??0012	Near CSEG:	01E8
??0013	Near CSEG:	01EF

??0014	Near CSEG:01FC
??0015	Near CSEG:0225
??0016	Near CSEG:0245
??0017	Near CSEG:025C
??0018	Near CSEG:0285
??0019	Near CSEG:028D
??001A	Near CSEG:029A
??001B	Near CSEG:02EE
??001C	Near CSEG:0471
??001D	Near CSEG:048E
??001E	Near CSEG:0497
??001F	Near CSEG:0316
??0020	Near CSEG:0336
??0021	Near CSEG:034D
??0022	Near CSEG:0376
??0023	Near CSEG:037E
??0024	Near CSEG:038B
??0025	Near CSEG:039C
??0026	Near CSEG:03AB
??0027	Near CSEG:03B7
??0028	Near CSEG:03E6
??0029	Near CSEG:040D
??002A	Near CSEG:0414
??002B	Near CSEG:0421

??002C	Near CSEG:0448
??002D	Near CSEG:044F
??002E	Near CSEG:045C
??DATE	Text "05/22/23"
??FILENAME	Text "lab5-3 "
??TIME	Text "14:31:38"
??VERSION	Number 0205
@CPU	Text 0D0FH
@CURSEG	Text CSEG
@FILENAME	Text LAB5-3

Symbol Table

@WORDSIZE	Text	4
ARRAY	Word	DSEG:0099
DIGIT	Word	DSEG:0028
END_PROGRAM	Near	CSEG:04AE
ERROR_MSG	Byte	DSEG:002B
EXISTS	Byte	DSEG:2102
FIND_VALUE_MSG	Byte	DSEG:209D
I	Word	DSEG:2099
INNER_BRACKETS	Byte	DSEG:001C
INPUT_MSG	Byte	DSEG:0076
INPUT_NUMBER	Byte	DSEG:0004
IS_ERROR	Byte	DSEG:002A
IS_NEGATIVE	Byte	DSEG:0025
J	Word	DSEG:209B
LAST_BRACKET	Byte	DSEG:001F
M	Word	DSEG:0002
MAIN	Near	CSEG:0000
MATRIX_MSG	Byte	DSEG:0014
M_MSG	Byte	DSEG:0053
N	Word	DSEG:0000
NOT_FOUND_MSG	Byte	DSEG:20C2

NUMBER	Word	DSEG:0026
N_MSG	Byte	DSEG:0034
RESTART_M	Near	CSEG:04A4
RESTART_N	Near	CSEG:049A
REST_M	Near	CSEG:00C0
REST_N	Near	CSEG:0005
RETRY_MSG	Byte	DSEG:20DB
THE_END	Near	CSEG:0468
TMP	Word	DSEG:2103

Macro Name

ENTER_MATRIX

FIND_ELEMENT

MOV_NUMBER

OUTPUT_FOUND_VALUES

OUTPUT_MESSAGE

OUTPUT_RESULT

READ

TRANSFORM

Groups & Segments

Bit Size Align Combine Class

CSEG

16 04B3 Para

Public CODE

DSEG	16 2105 Para	Public DATA	
STSEG	16 0140 Para	Stack	STACK

Схема функціонування програм.

Програма 1.

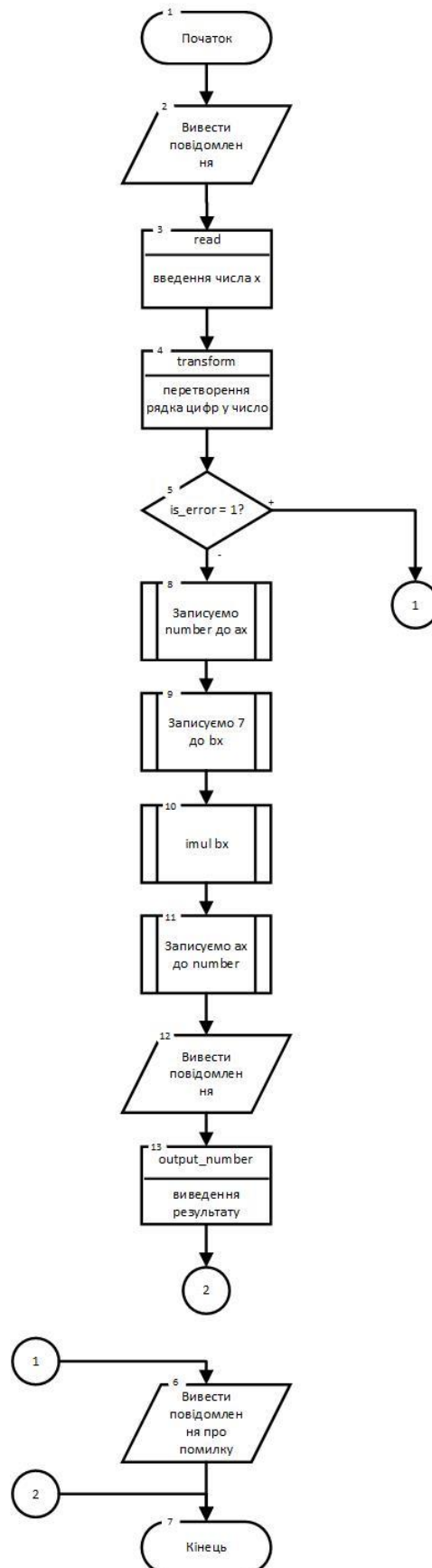


Рисунок 1. Схема основної частини програми.



Рисунок 2. Схема процедури read

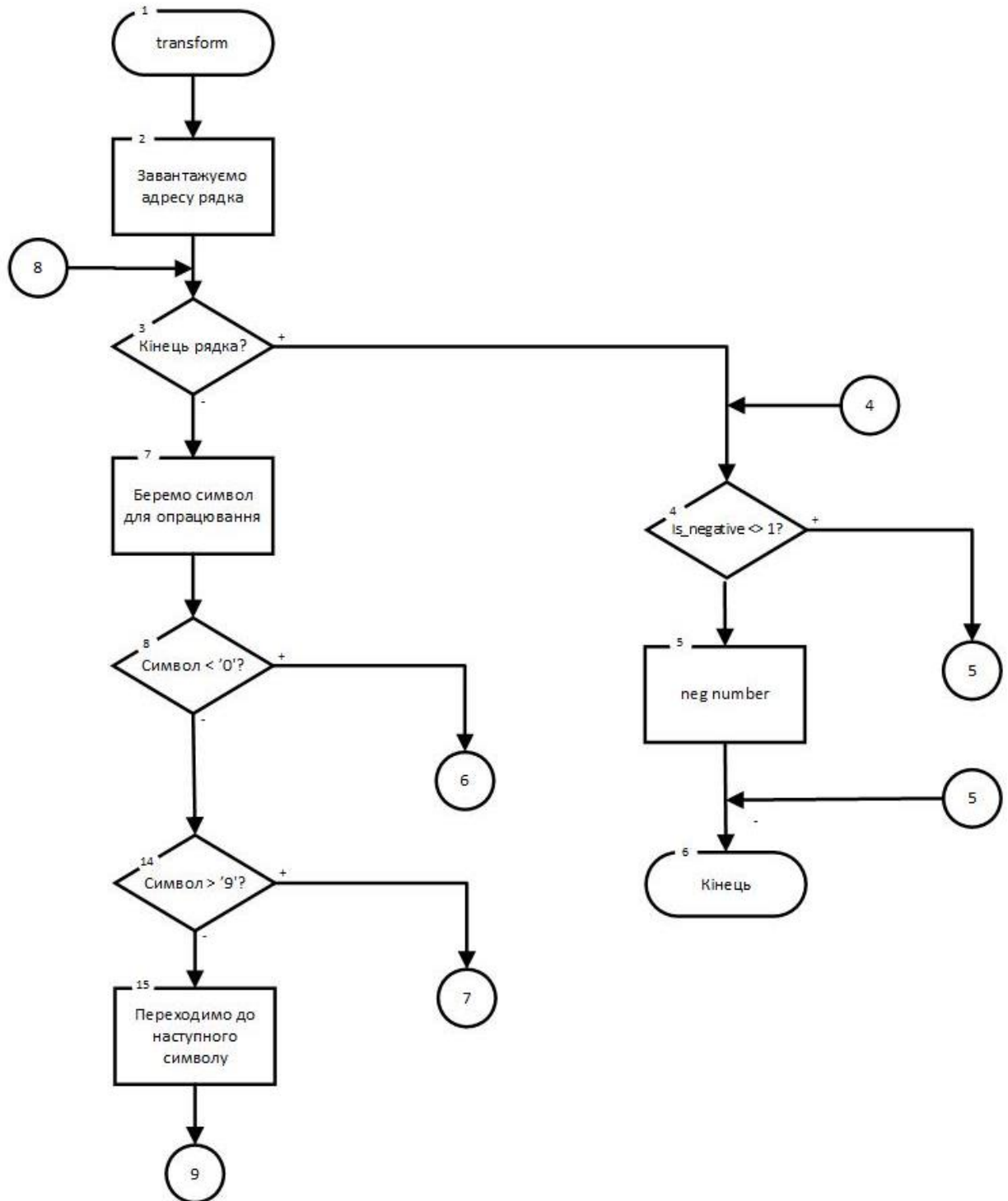


Рисунок 3. Схема процедури transform

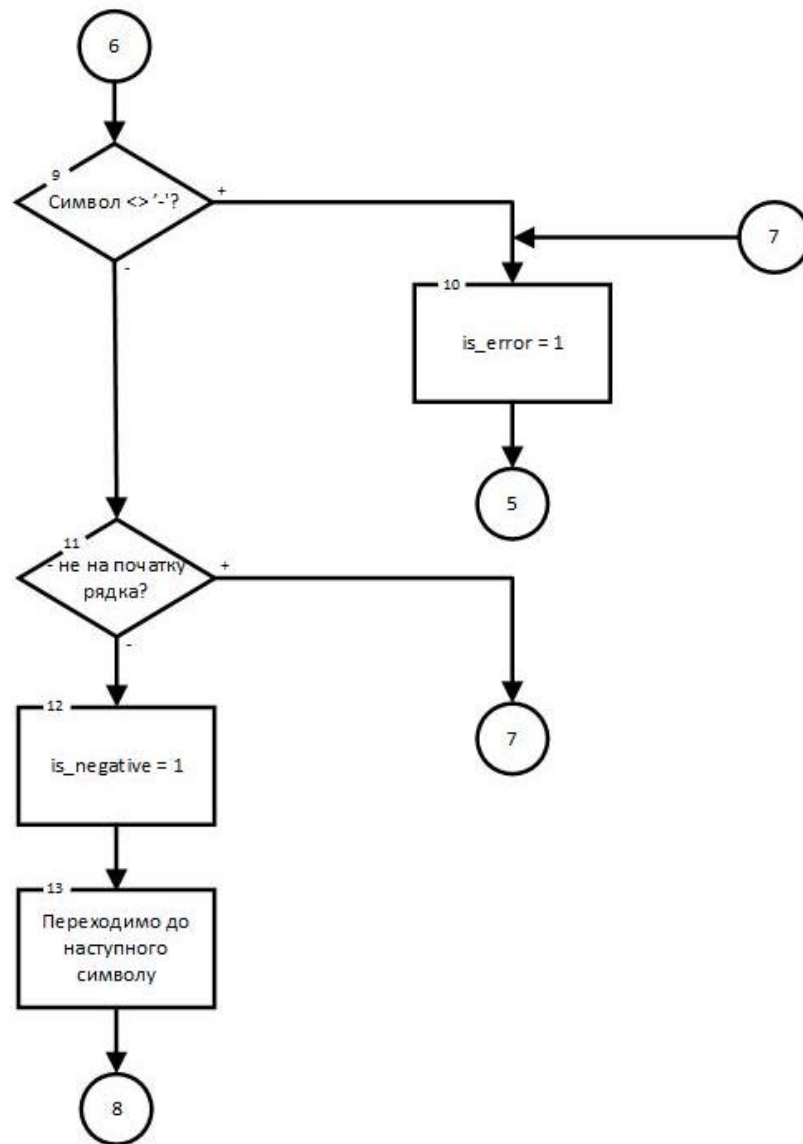


Рисунок 4. Схема процедури transform (продовження 1)

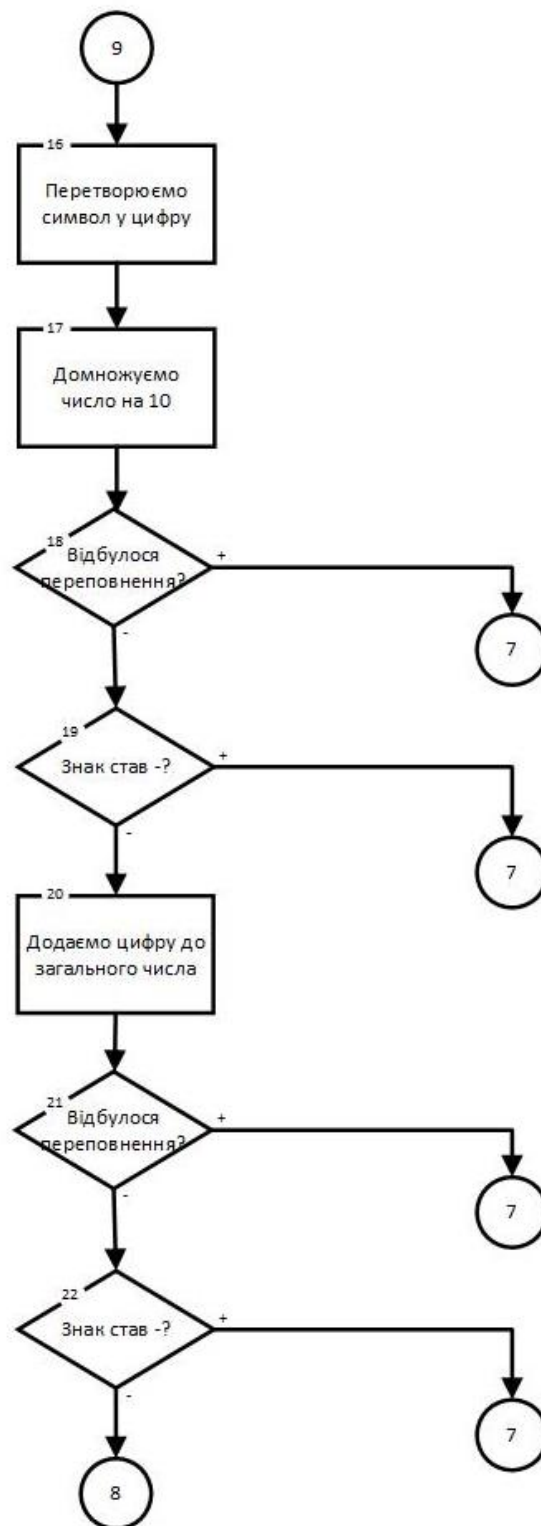


Рисунок 5. Схема процедури transform (продовження 2)

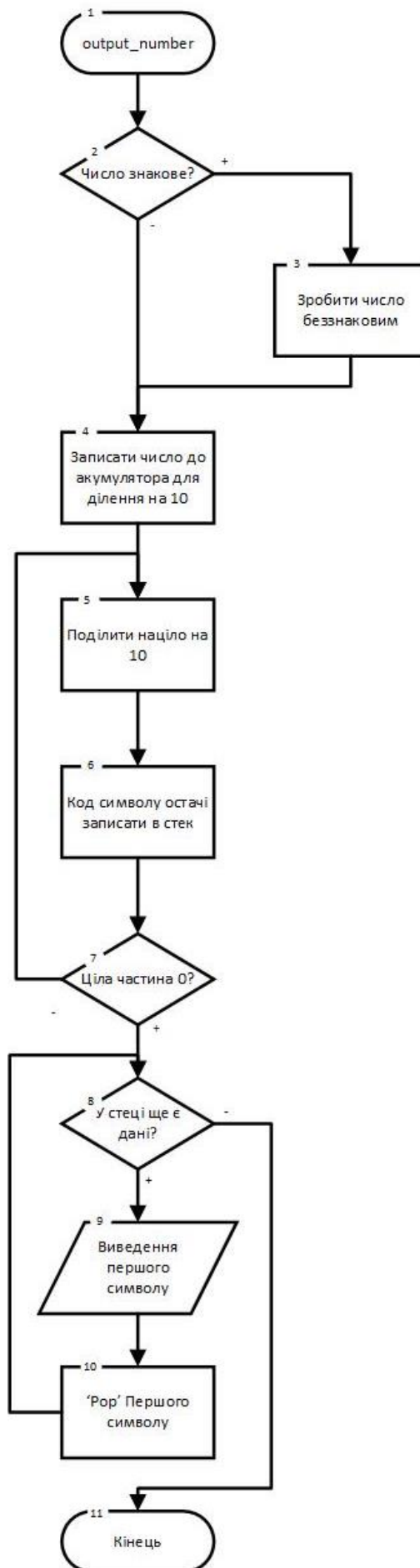


Рисунок 6. Схема макросу output_number

Програма 2.

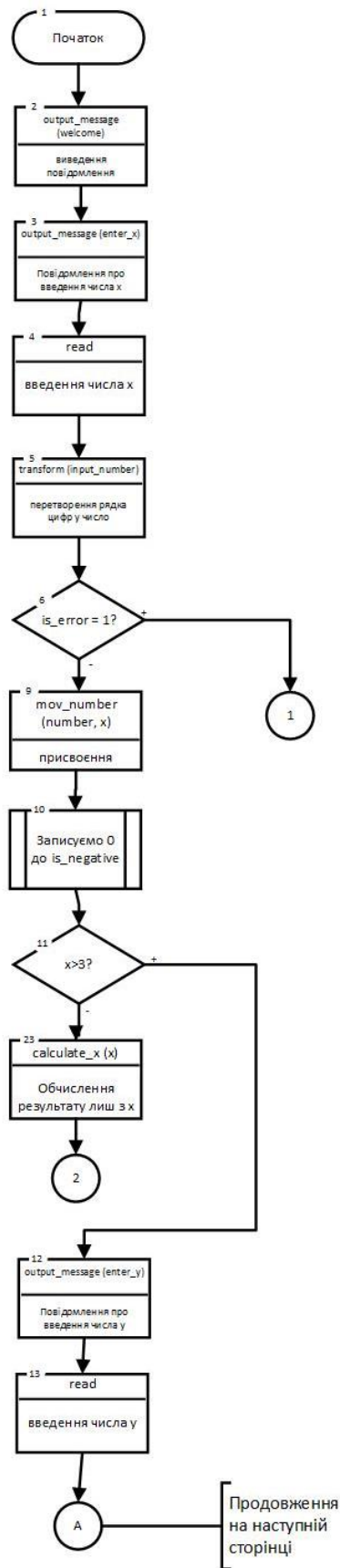


Рисунок 7. Схема основної частини програми

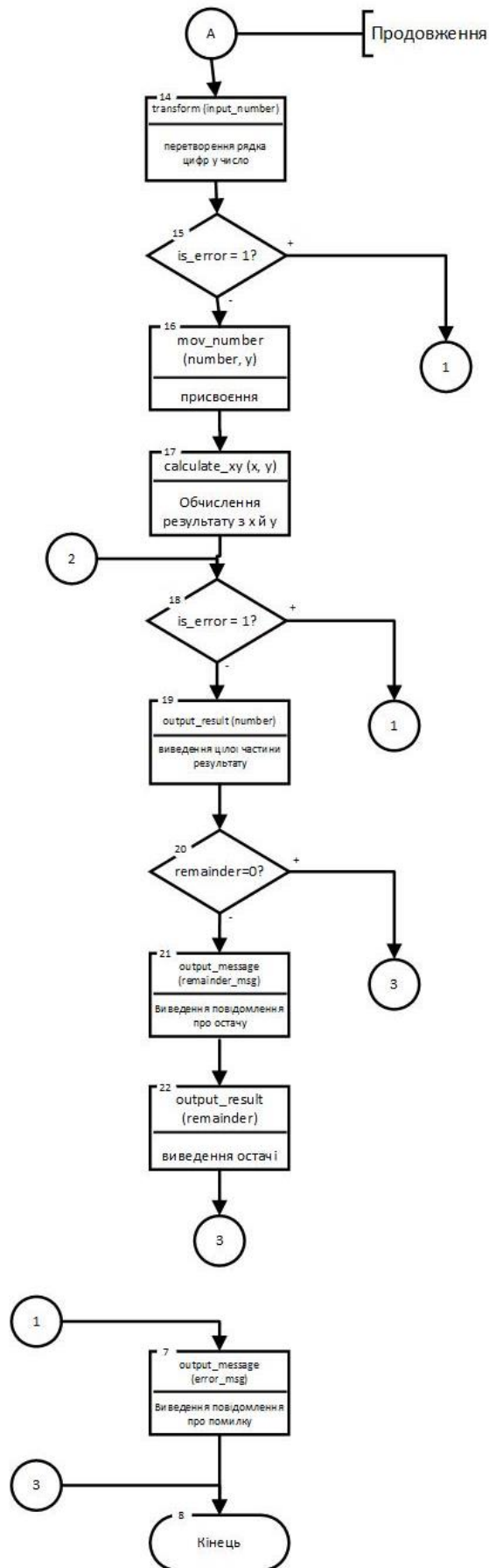


Рисунок 8. Схема основної частини програми (продовження)



Рисунок 9. Схема макросу output_message



Рисунок 10. Схема макросу mov_number



Рисунок 11. Схема макросу read

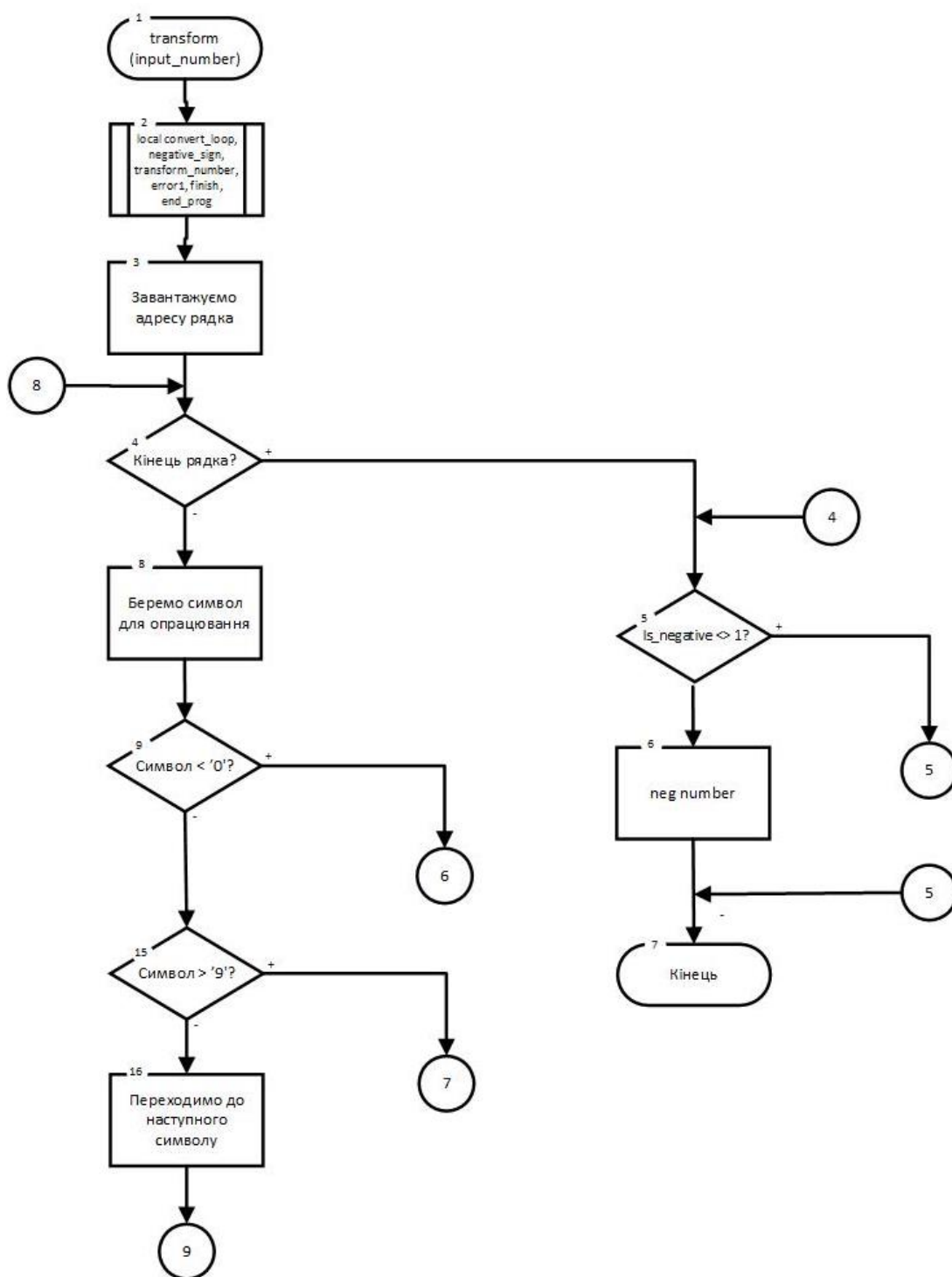


Рисунок 12. Схема макросу transform

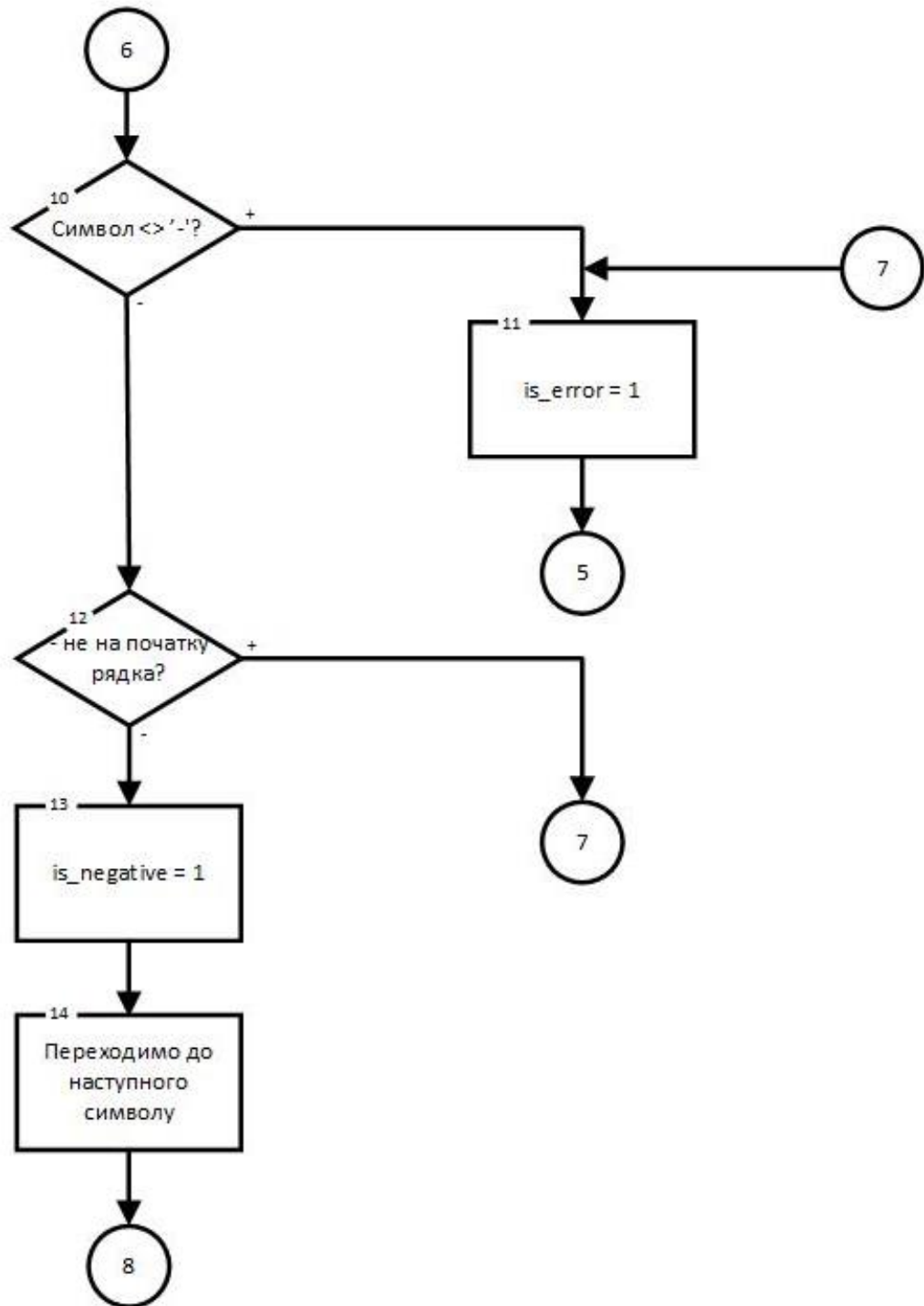


Рисунок 13. Схема макросу transform (продовження 1)

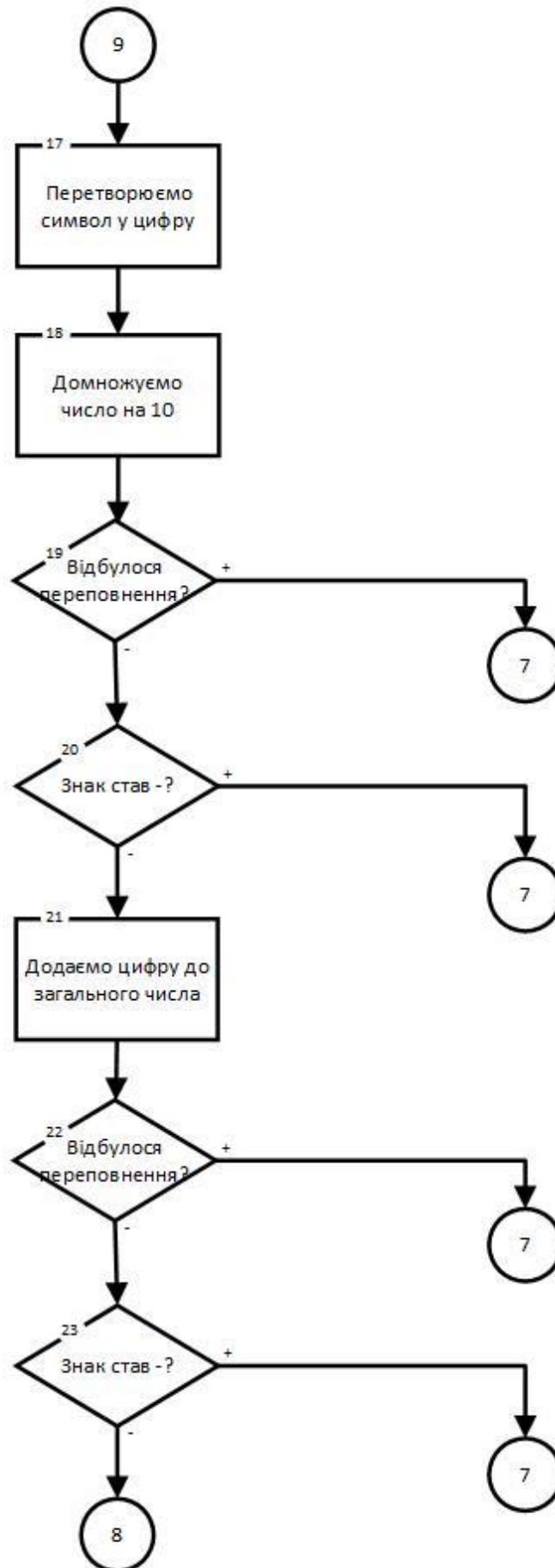


Рисунок 14. Схема макросу transform (продовження 2)

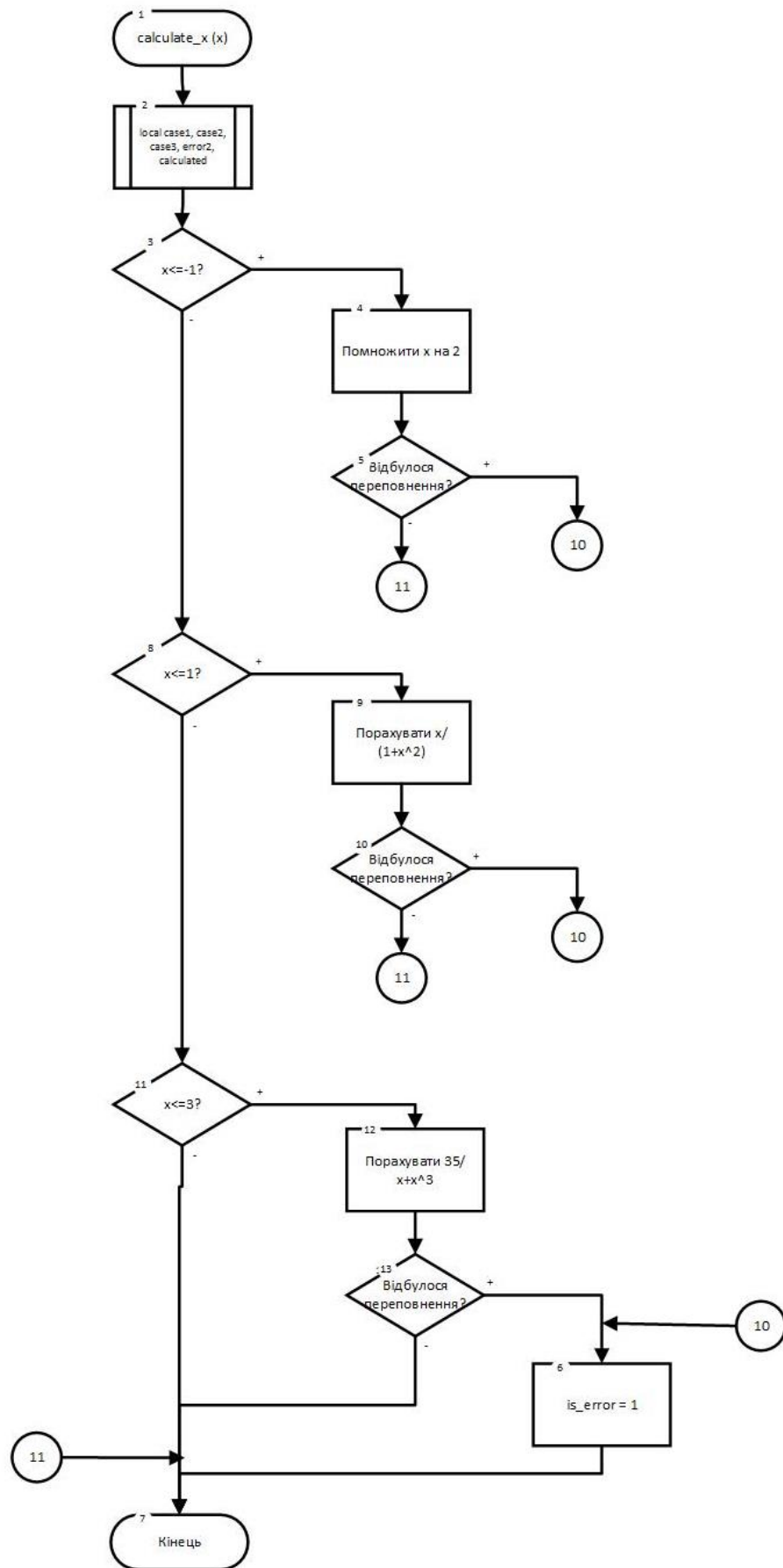


Рисунок 15. Схема макросу calculate_x

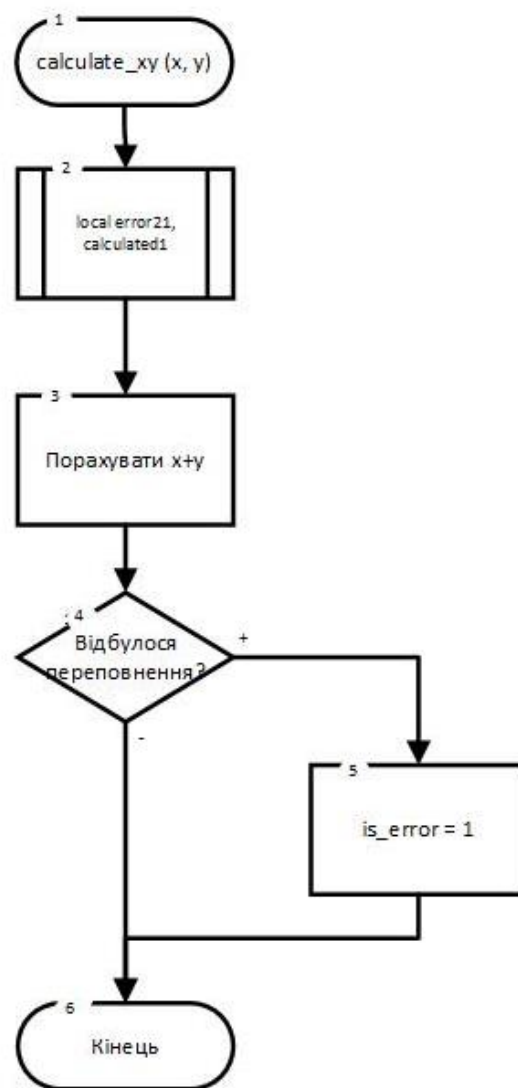


Рисунок 16. Схема макросу calculate_xy



Рисунок 17. Схема макросу output_result

Програма 3.

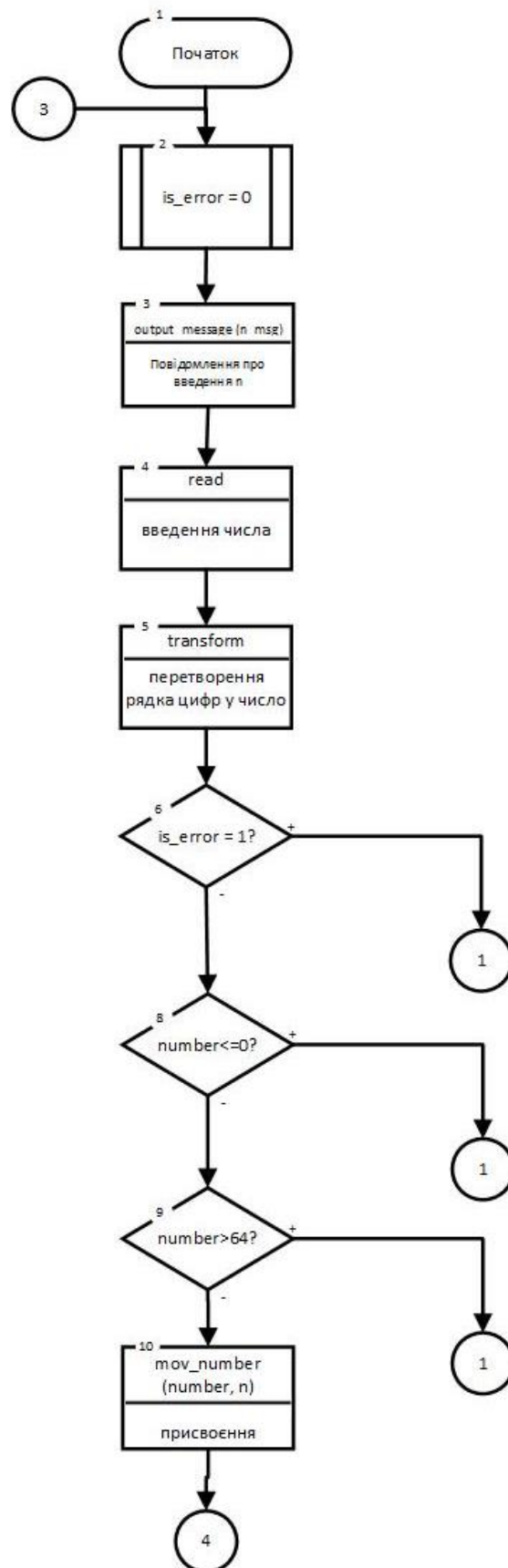


Рисунок 18. Схема основної частини програми

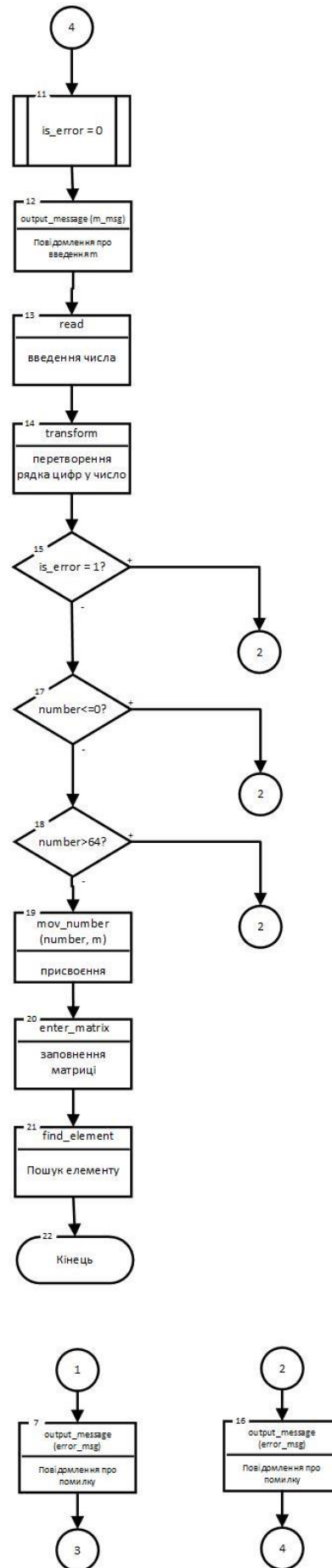


Рисунок 19. Схема основної частини програми (продовження)



Рисунок 20. Схема макросу output_message



Рисунок 21. Схема макросу mov_number



Рисунок 22. Схема макросу read

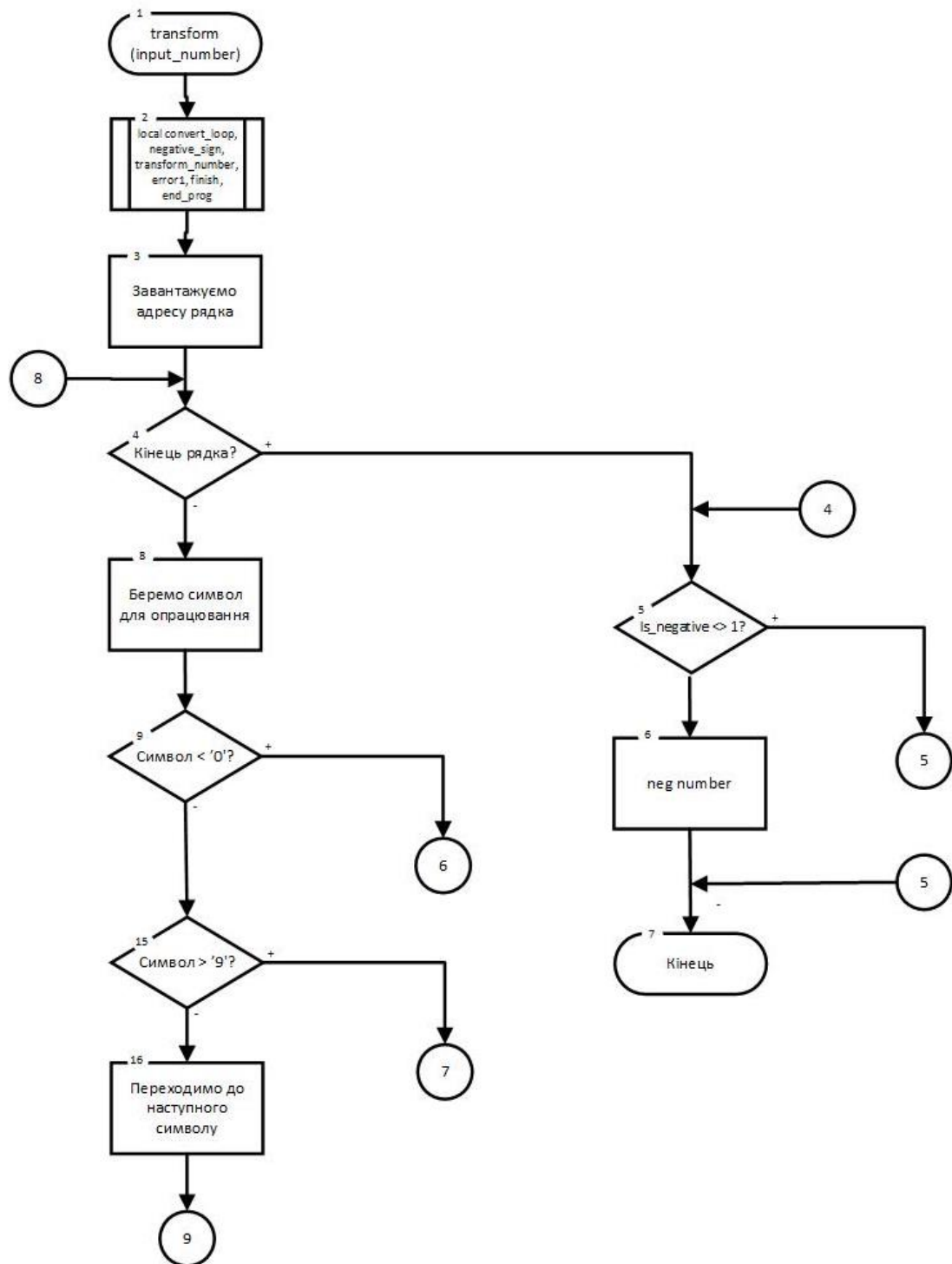


Рисунок 23. Схема макросу transform

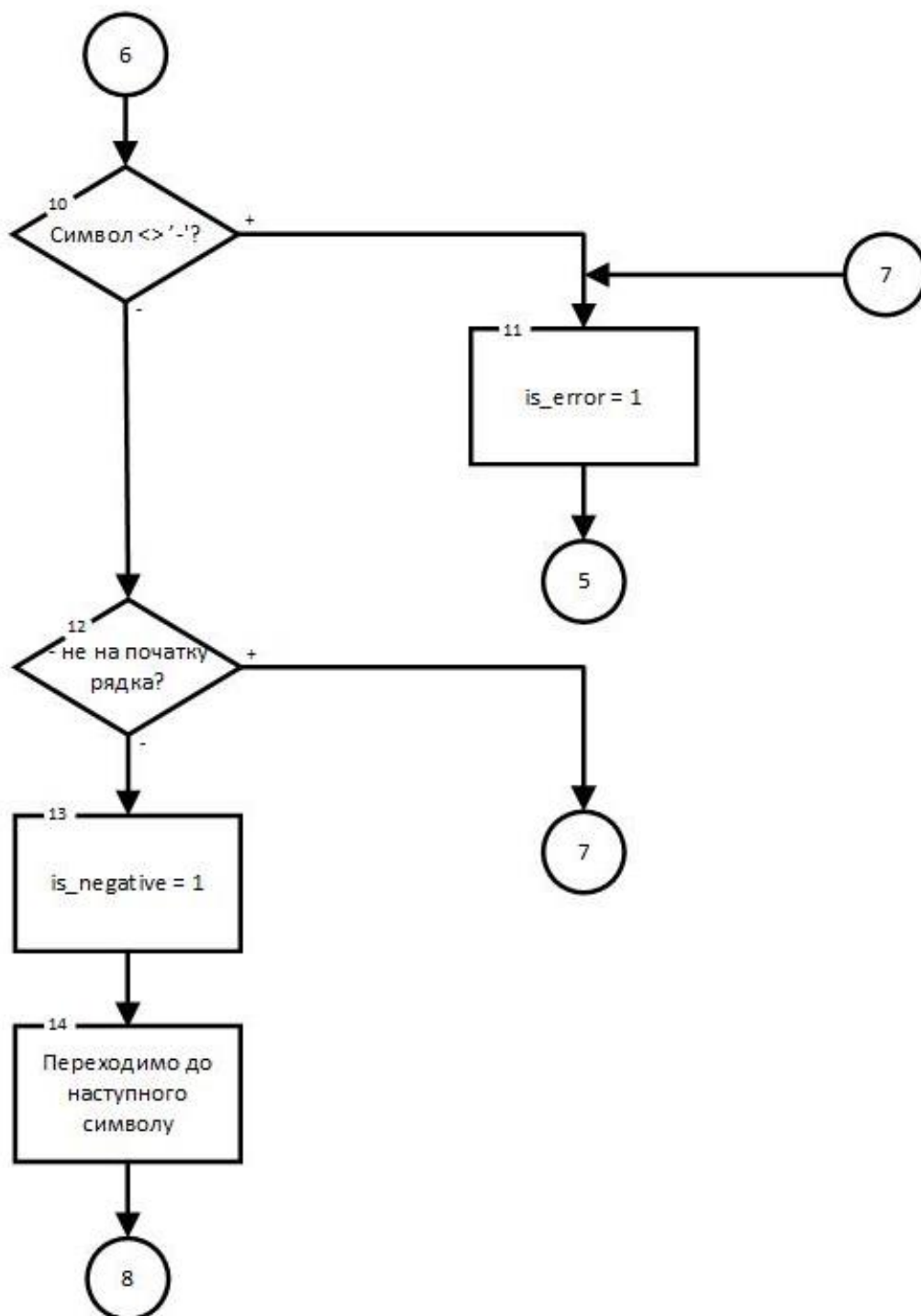


Рисунок 24. Схема макросу transform (продовження 1)

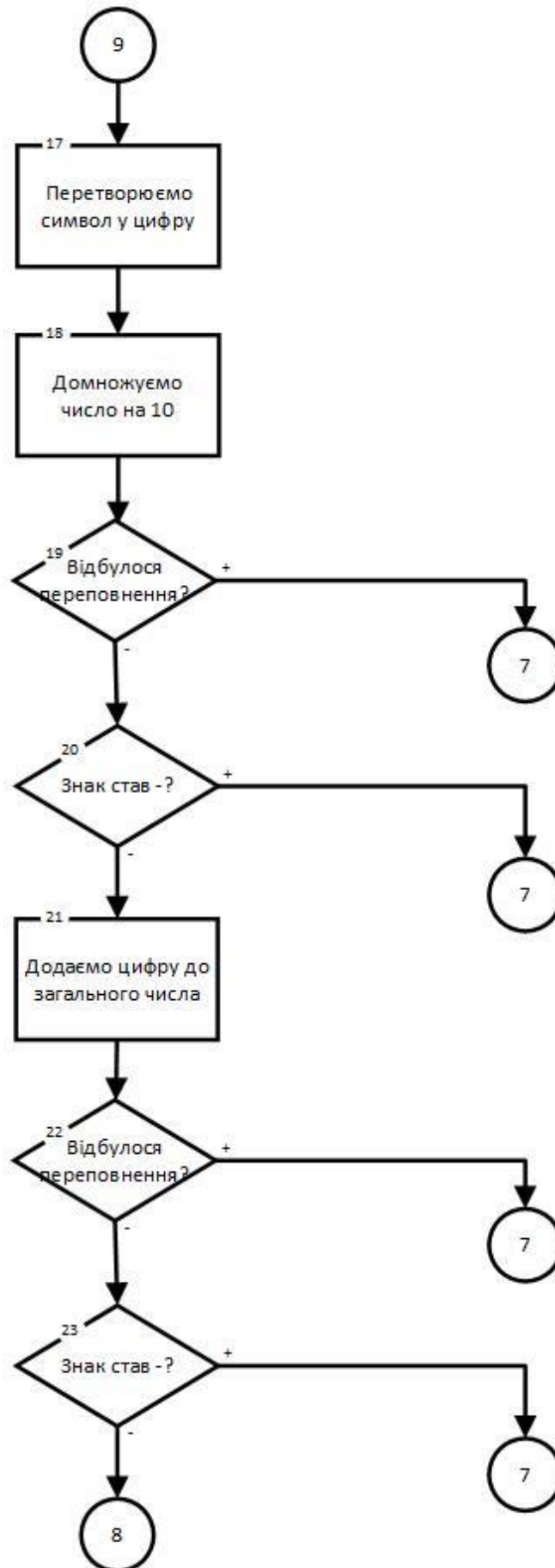


Рисунок 25. Схема макросу transform (продовження 2)



Рисунок 26. Схема макросу output_result

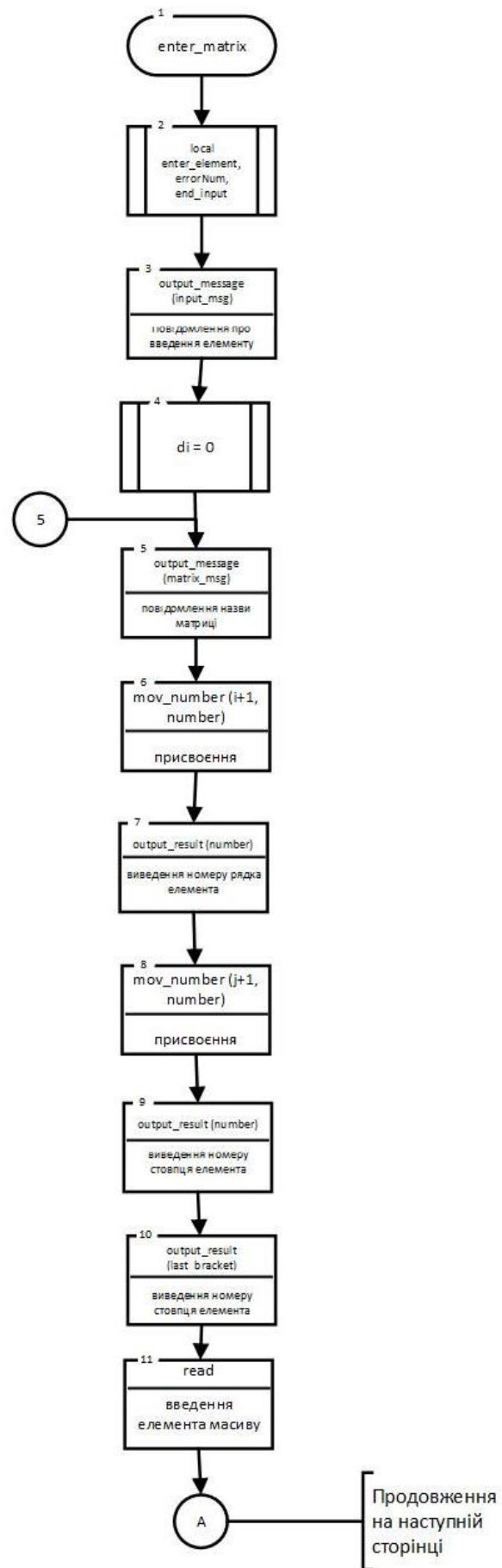
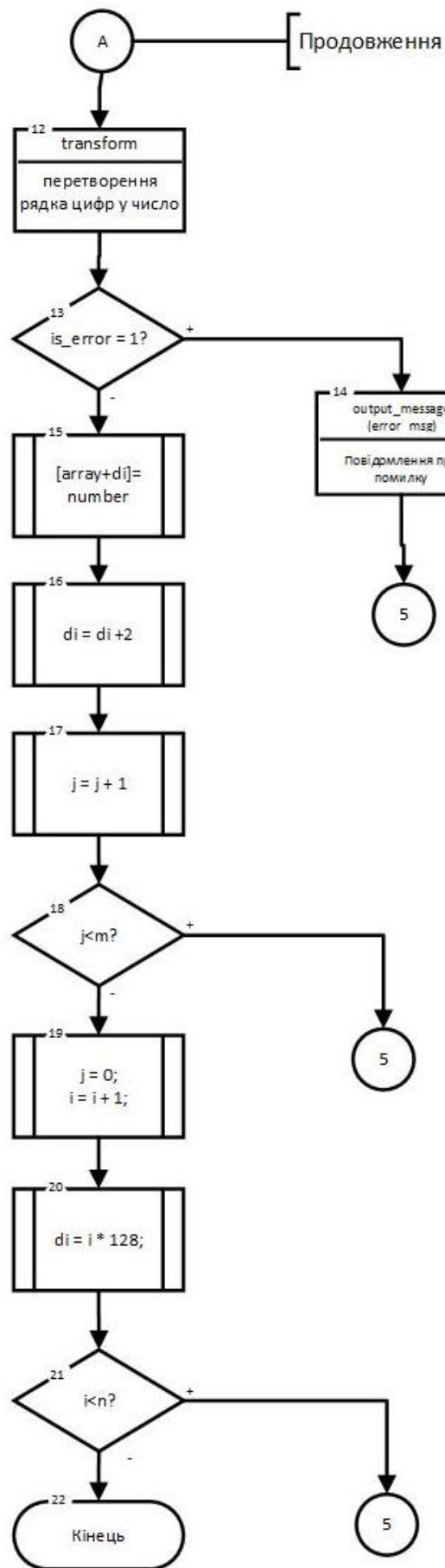


Рисунок 27. Схема макросу enter_matrix

Рисунок 28. Схема макросу `enter_matrix` (продовження)

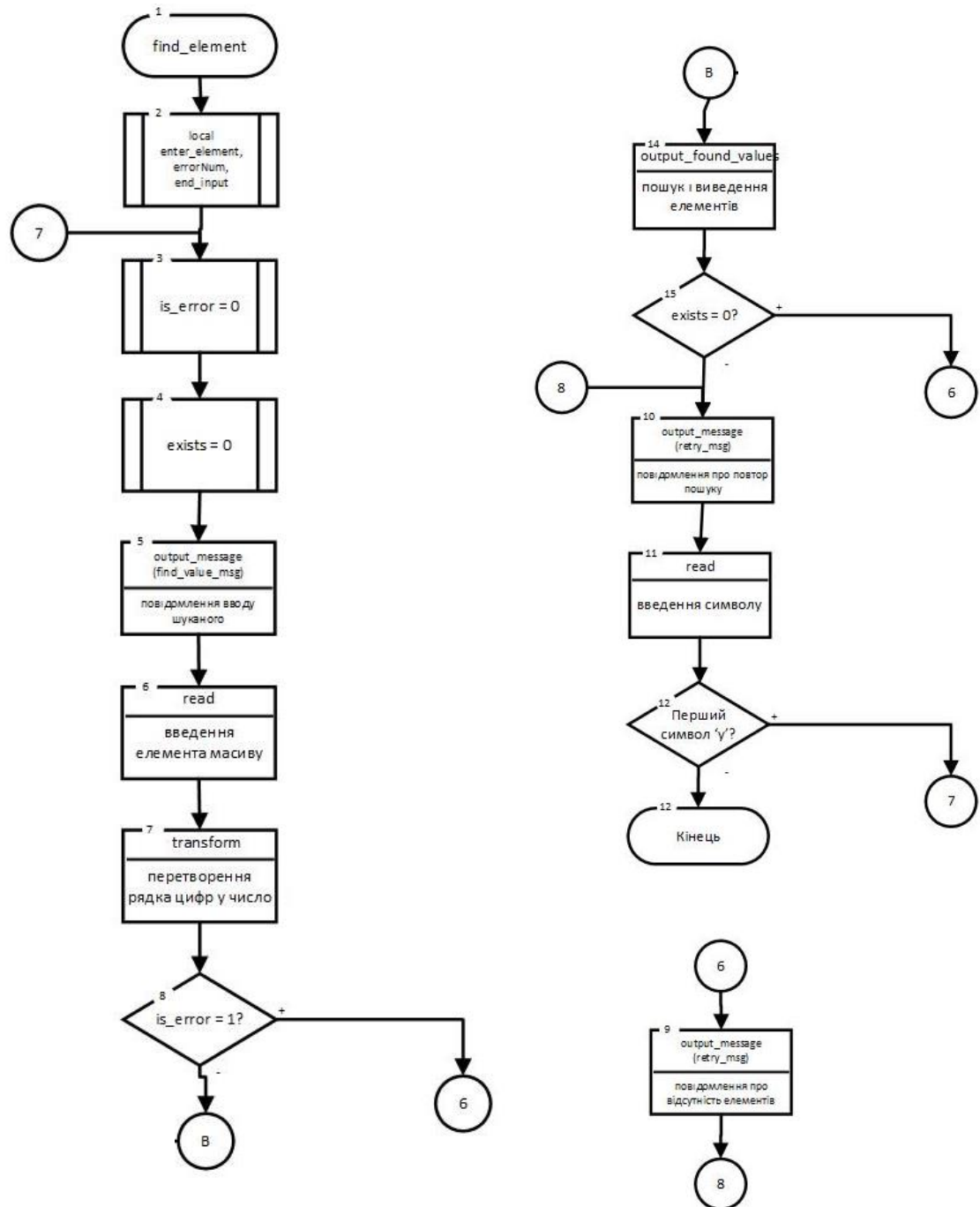


Рисунок 29. Схема макросу find_element

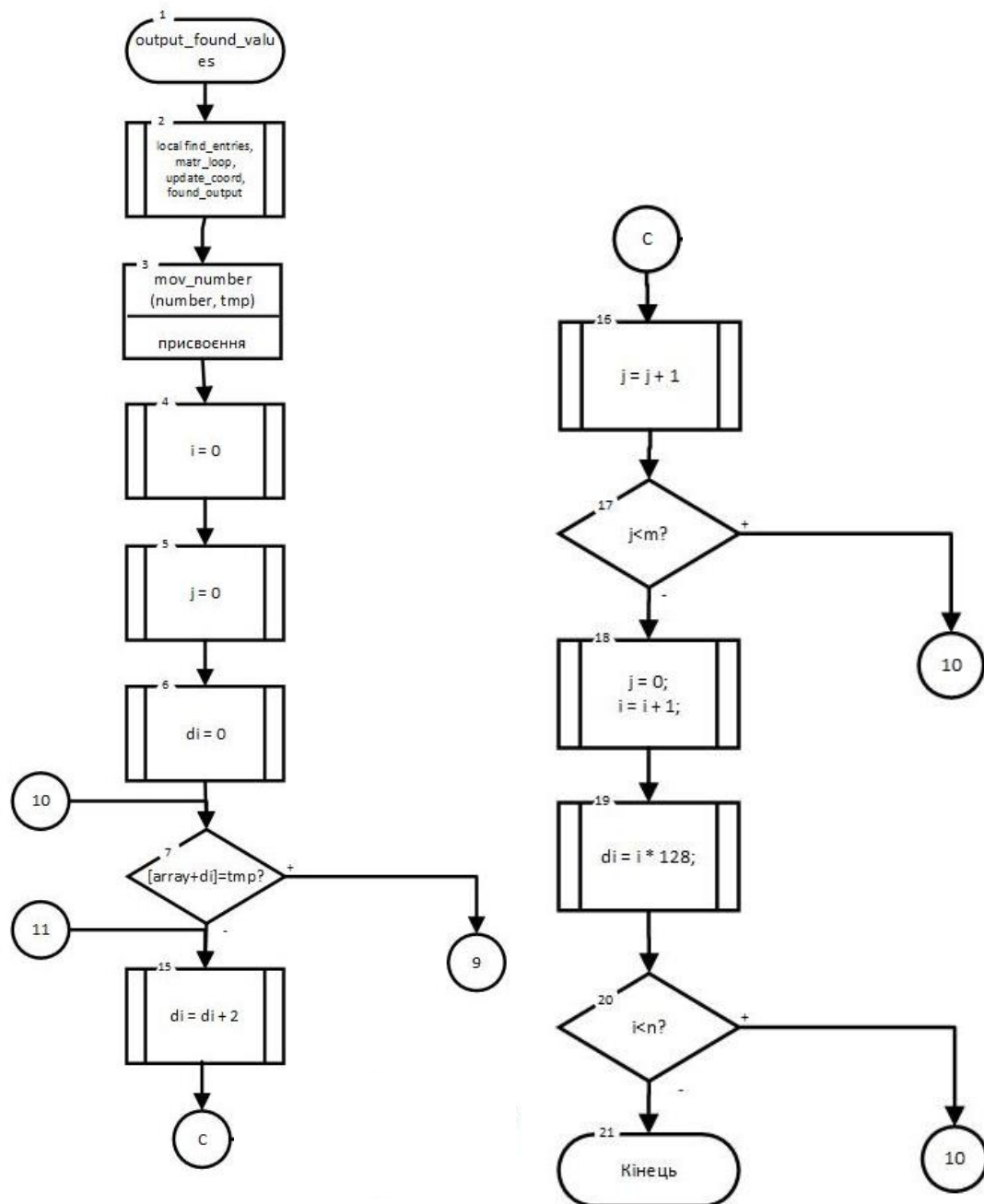


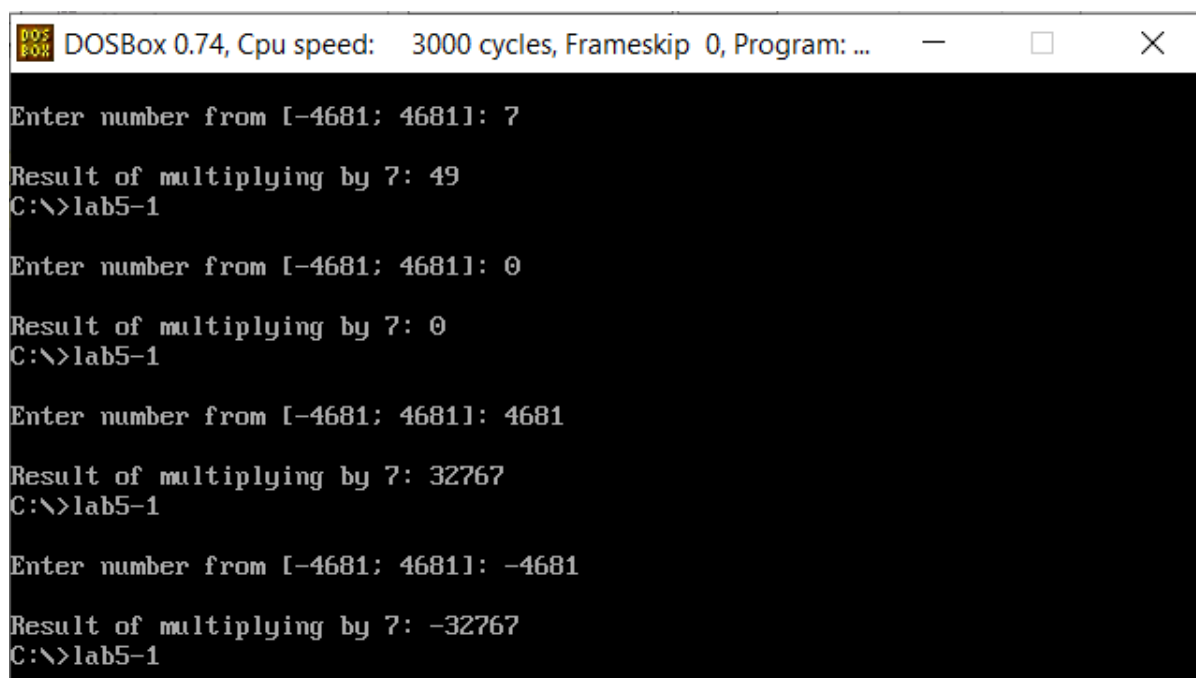
Рисунок 30. Схема макросу output_found_values



Рисунок 31. Схема макросу `output_found_values` (продовження)

Приклад виконання програми.

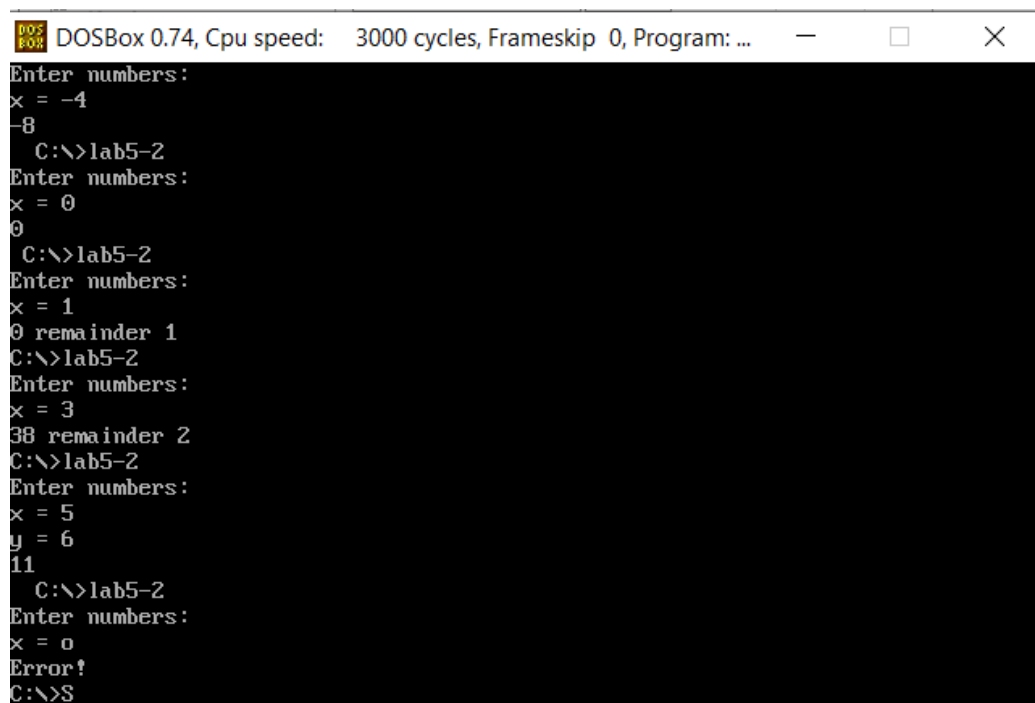
Програма 1.



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Enter number from [-4681; 4681]: 7
Result of multiplying by 7: 49
C:\>lab5-1
Enter number from [-4681; 4681]: 0
Result of multiplying by 7: 0
C:\>lab5-1
Enter number from [-4681; 4681]: 4681
Result of multiplying by 7: 32767
C:\>lab5-1
Enter number from [-4681; 4681]: -4681
Result of multiplying by 7: -32767
C:\>lab5-1
```

Рисунок 32. Приклад виконання програми 1.

Програма 2.



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Enter numbers:
x = -4
y = -8
C:\>lab5-2
Enter numbers:
x = 0
y = 0
C:\>lab5-2
Enter numbers:
x = 1
y = 0 remainder 1
C:\>lab5-2
Enter numbers:
x = 3
y = 38 remainder 2
C:\>lab5-2
Enter numbers:
x = 5
y = 6
11
C:\>lab5-2
Enter numbers:
x = 0
Error!
C:\>S
```

Рисунок 33. Приклад виконання програми 2.

Програма 3.

```
C:\>lab5-3
Enter number of rows (1-64): 2
Enter number of columns (1-64): 2
Enter elements ([-32767; 32767])
matr[1][1] => 1
matr[1][2] => 2
matr[2][1] => 3
matr[2][2] => 1
What value do you want to find? : 1
matr[1][1]
matr[2][2]
Continue the search? ('y' for yes): y
What value do you want to find? : 6
Element is not found.
Continue the search? ('y' for yes): y
What value do you want to find? : 2
matr[1][2]
Continue the search? ('y' for yes): y
What value do you want to find? : 0
Element is not found.
Continue the search? ('y' for yes): n
C:\>S
```

Рисунок 34. Приклад виконання програми 3.

Висновок.

Отже, у даній роботі я ознайомився із застосування макрозасобів мови асемблер.

Було складено програми на нижче наведені завдання:

- переписати програму комп'ютерного практикуму №2 з використанням одного макросу;
- переписати програму комп'ютерного практикуму №3 з використанням макросів та передачею параметрів в них;
- переписати програму з двовимірним масивом комп'ютерного практикуму № 4 з використанням макросів та залученням міток в тілі макросу.

Програми було скомпільовано, налагоджено та виконано. У результаті виконання програм отримується коректний й очікуваний результат.