

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 № 4 з дисципліни

«Системне програмне забезпечення»

«Робота з масивами»

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

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

Київ 2023

Комп'ютерний практикум 4

Робота з масивами

Постановка завдання.

1. Написати програму, яка повинна мати наступний функціонал:
 - a. Можливість введення користувачем розміру одномірного масиву.
 - b. Можливість введення користувачем значень елементів одномірного масиву.
 - c. Можливість знаходження суми елементів одномірного масиву.
 - d. Можливість пошуку максимального (або мінімального) елемента одномірного масиву.
 - e. Можливість сортування одномірного масиву цілих чисел загального вигляду.
2. Написати програму, яка буде мати наступний функціонал:
 - a. Можливість введення користувачем розміру двомірного масиву.
 - b. Можливість введення користувачем значень елементів двомірного масиву.
 - c. Можливість пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий елемент вводять користувач.
3. Програма повинна мати захист від некоректного введення вхідних даних (символи, переповнення і т.і.)

Хід роботи.**Текст програми 1.**

```
STSEG SEGMENT PARA STACK "STACK"
```

```
DB 64 DUP("STACK")
```

```
STSEG ENDS
```

```
DSEG SEGMENT PARA PUBLIC "DATA"
```

```
arr_size dw 0
```

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

```
array_message db "array[$"
```

```
close_bracket db "]" => "$"
```

```
is_negative db 0
```

```
number dw 0
```

```
digit dw 0
```

```
is_error db 0
```

```
error_msg db "Error!$"
```

```
size_msg db "Enter the size of array (1-16) => $"
```

```
input_msg db "Enter elements' values in range [-2047; 2047]$"
```

```
array dw 16 dup (?)
```

```
sum_msg db "Sum of array elements = $"
```

```
sum dw 0
```

```
min_msg db 13,10, "Min array element = $"
```

```
min dw 0
```

```
max_msg db 13,10, "Max array element = $"
```

```
max dw 0
```

```
sorted_msg db 13, 10, "Sorted array: $"
```

```
counter dw 0
```

```
DSEG ENDS
```

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

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

```
main proc
```

```
    mov ax, dseg
```

```
    mov ds, ax
```

```
start:
```

```
    mov is_error, 0
```

```
    lea dx, size_msg
```

```
    mov ah, 9
```

```
    int 21h
```

```
    call read
```

```
    call transform
```

```
    cmp is_error, 1
```

```
    je error_retry
```

```
    cmp number, 0
```

```
    jle error_retry
```

cmp number, 16

jg error_retry

mov ax, number

mov arr_size, ax

call enter_array

cmp is_error, 1

je start

call calculate_sum

mov ax, sum

mov number, ax

call output_result

call minmax_find

lea dx, min_msg

mov ah, 9

int 21h

mov ax, min

mov number, ax

call output_result

```
lea dx, max_msg
```

```
mov ah, 9
```

```
int 21h
```

```
mov ax, max
```

```
mov number, ax
```

```
call output_result
```

```
call sorting
```

```
call output_array
```

```
jmp end_program
```

```
error_retry:
```

```
    LEA dx, error_msg
```

```
    MOV ah,9
```

```
    INT 21h
```

```
    jmp start
```

```
end_program:
```

```
    mov AH, 4CH
```

```
    int 21H
```

```
    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 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

ret

transform endp

output_result proc

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

                ret

output_result endp


enter_array proc

                lea dx, input_msg
                mov ah, 9
                int 21h


;новый рядок

                mov al,10
                int 29h
                mov al,13
                int 29h


                mov counter, 0
                mov di, 0


enter_element:
```

```
lea dx, array_message
```

```
mov ah, 9
```

```
int 21h
```

```
mov ax, counter
```

```
mov number, ax
```

```
add number, 1
```

```
call output_result
```

```
lea dx, close_bracket
```

```
mov ah, 9
```

```
int 21h
```

```
call read
```

```
call transform
```

```
cmp is_error, 1
```

```
je errorNum
```

```
cmp number, -2047
```

```
jle errorNum
```

```
cmp number, 2047
```

```
jge errorNum
```

```
mov ax, number
```

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

```
inc counter

add di, 2

mov cx, counter

cmp cx, arr_size

jne enter_element

jmp end_input
```

errorNum:

```
mov is_error, 0

LEA dx, error_msg

MOV ah,9

INT 21h

jmp enter_element
```

end_input:

```
ret
```

enter_array endp

calculate_sum proc

```
lea dx, sum_msg

mov ah, 9

int 21h

xor dx, dx

xor ax, ax
```

```
mov cx, arr_size
```

```
mov di, 0
```

```
sum_loop:
```

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

```
    add dx, ax
```

```
    add di, 2
```

```
    loop sum_loop
```

```
mov sum, dx
```

```
ret
```

```
calculate_sum endp
```

```
minmax_find proc
```

```
    mov ax, [array]
```

```
    mov min, ax
```

```
    mov max, ax
```

```
    mov cx, arr_size
```

```
    sub cx, 1
```

```
    jcxz exit
```

```
    mov di, 2
```

```
read_num:
```

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

```
    cmp ax, min
```

```
    jl min_found  
  
    cmp ax, max  
  
    jg max_found  
  
    add di, 2  
  
    loop read_num  
  
    jmp exit
```

min_found:

```
    mov min, ax  
  
    add di, 2  
  
    dec cx  
  
    jcxz exit  
  
    jmp read_num
```

max_found:

```
    mov max, ax  
  
    add di, 2  
  
    dec cx  
  
    jcxz exit  
  
    jmp read_num
```

exit:

```
    ret
```

minmax_find endp

sorting proc

 cmp arr_size, 1

 je sort_done

 mov si, 0

outer_loop:

 mov cx, arr_size

 dec cx

 mov di, 0

inner_loop:

 mov ax, [array+di]

 cmp ax, [array+di+2]

 jng continue

 mov dx, [array+di]

 mov ax, [array+di+2]

 mov [array+di], ax

 mov [array+di+2], dx

continue:

 add di, 2

 loop inner_loop

 inc si

 cmp si, arr_size

 jl outer_loop

sort_done:

ret

sorting endp

output_array proc

lea dx, sorted_msg

mov ah, 09h

int 21h

mov di, 0

mov si, 0

show:

mov dx, [array+di]

mov number, dx

call output_result

mov al, ''

int 29h

add di, 2

inc si

cmp si, arr_size

jne show

ret

output_array endp

CSEG ENDS

end main

Текст програми 2.

```
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

    mov ax, dseg
    mov ds, ax

    rest_n:

    mov is_error, 0

    lea dx, n_msg
    mov ah, 9
    int 21h

    call read
    call transform
    cmp is_error, 1
    je restart_n
    cmp number, 0
    jle restart_n
```

cmp number, 64

jg restart_n

mov ax, number

mov n, ax

rest_m:

mov is_error, 0

lea dx, m_msg

mov ah, 9

int 21h

call read

call transform

cmp is_error, 1

je restart_m

cmp number, 0

jle restart_m

cmp number, 64

jg restart_m

mov ax, number

mov m, ax

call enter_matrix

call find_element

jmp end_program

restart_n:

LEA dx, error_msg

MOV ah,9

INT 21h

jmp rest_n

restart_m:

LEA dx, error_msg

MOV ah,9

INT 21h

jmp rest_m

end_program:

mov AH, 4CH

int 21H

ret

main endp

read proc

```

lea dx, input_number

mov ah, 10

int 21h

ret

read endp

```

```

transform proc

    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 done
```

finish:

```
    cmp is_negative, 1  
    jne done  
    neg number
```

done:

```
    xor ch, ch  
    ret
```

transform endp

output_result proc

```
    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

ret

```
output_result endp
```

```
enter_matrix proc
```

```
    lea dx, input_msg
```

```
    mov ah, 9
```

```
    int 21h
```

```
    mov di, 0
```

```
enter_element:
```

```
    lea dx, matrix_msg
```

```
    mov ah, 9
```

```
    int 21h
```

```
    mov ax, i
```

```
    mov number, ax
```

```
    add number, 1
```

```
    call output_result
```

```
    mov number, 0
```

```
    lea dx, inner_brackets
```

```
    mov ah, 9
```

```
    int 21h
```

```
    mov ax, j
```

```
    mov number, ax
```

```
add number, 1  
call output_result  
mov number, 0  
lea dx, last_bracket  
mov ah, 9  
int 21h
```

```
call read  
call transform  
cmp is_error, 1  
je errorNum
```

```
mov ax, number  
mov [array+di], ax
```

;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  
  
    LEA dx, error_msg  
  
    MOV ah,9  
  
    INT 21h  
  
    jmp enter_element
```

end_input:

```
    ret
```

enter_matrix endp

find_element proc

dialog_loop:

```
        mov is_error, 0  
  
        mov exists, 0
```

```
lea dx, find_value_msg
```

```
mov ah, 9
```

```
int 21h
```

```
call read
```

```
call transform
```

```
cmp is_error, 1
```

```
je not_found
```

```
call output_found_values
```

```
cmp exists, 0
```

```
je not_found
```

```
continue_dialog:
```

```
lea dx, retry_msg
```

```
mov ah, 9
```

```
int 21h
```

```
call read
```

```
mov dl, [input_number+2]
```

```
cmp dl, 'y'
```

```
je dialog_loop
```

```
jne end_proc
```

```
not_found:
```

```
lea dx, not_found_msg
```

```
mov ah, 9
```

```
int 21h
```

```
jmp continue_dialog
```

```
end_proc:
```

```
ret
```

```
find_element endp
```

```
output_found_values proc
```

```
mov ax, number
```

```
mov tmp, ax
```

```
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
```

```
ret
```

```
found_output:
```

```
    mov exists, 1

    lea dx, matrix_msg
```

```
    mov ah, 9
    int 21h

    mov ax, i
    mov number, ax
    add number, 1
    call output_result
    lea dx, inner_brackets
    mov ah, 9
    int 21h
    mov ax, j
    mov number, ax
    add number, 1
    call output_result
    mov al, ']'
    int 29h
    jmp update_coord

output_found_values endp

CSEG ENDS

end main
```


Схема функціонування програми.

Програма 1.

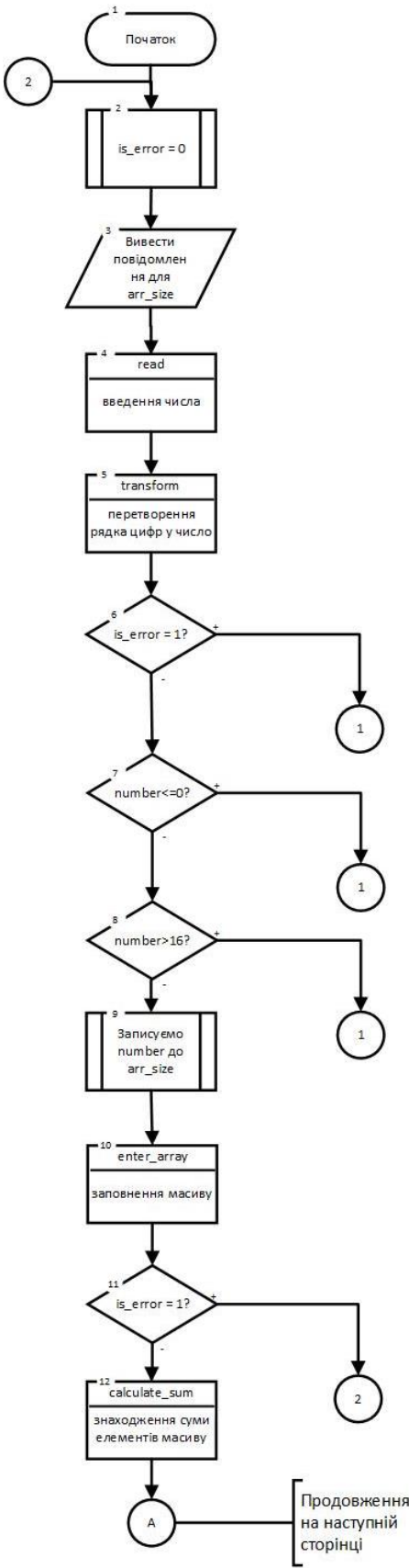


Рисунок 1. Схема основної частини програми.

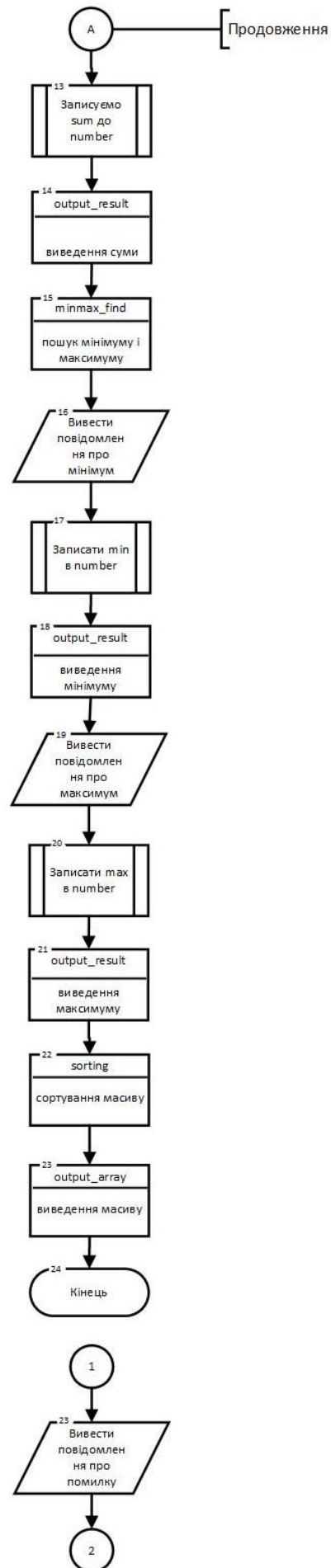


Рисунок 2. Схема основної частини програми (продовження).



Рисунок 3. Схема процедури read

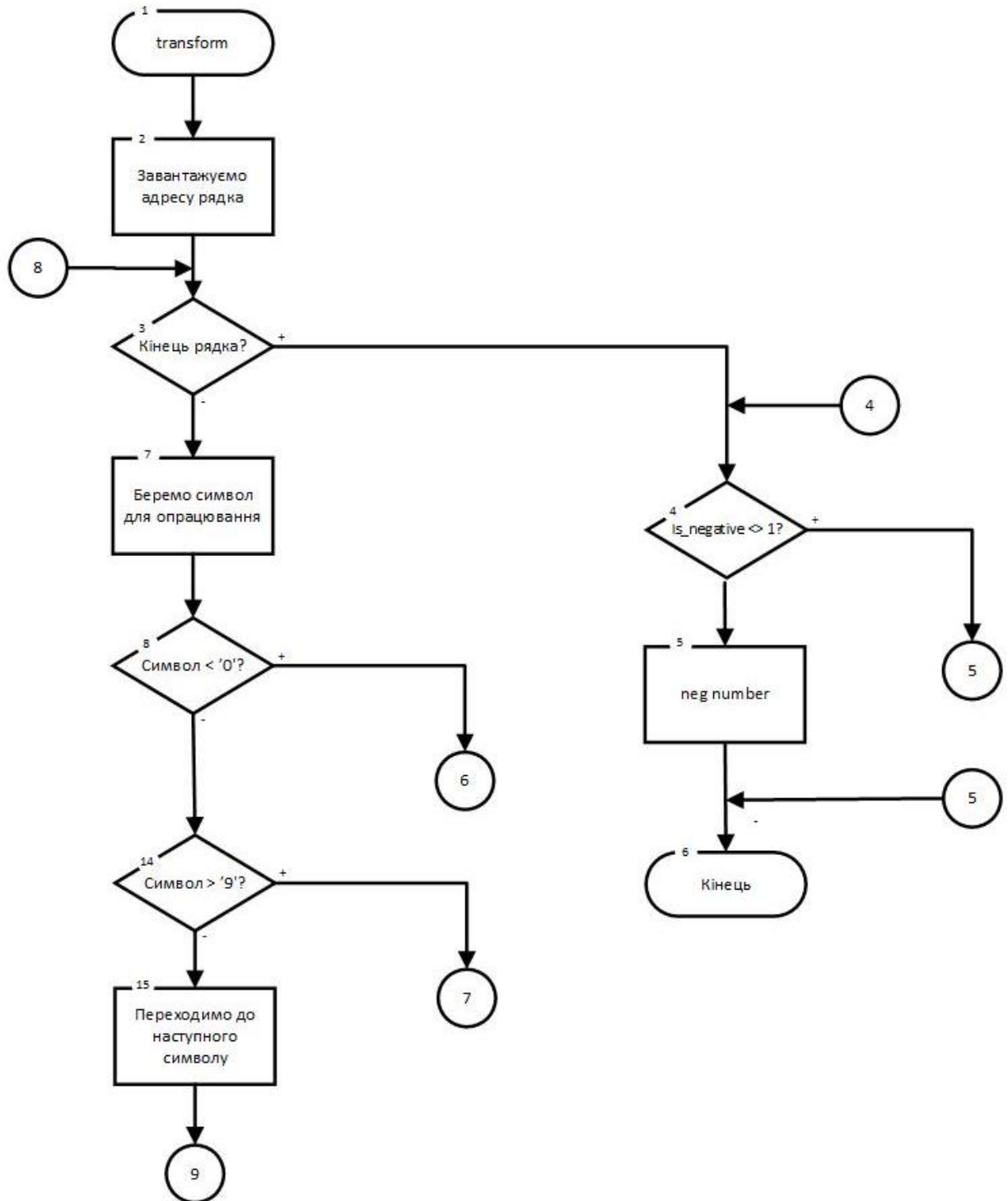


Рисунок 4. Схема процедури transform

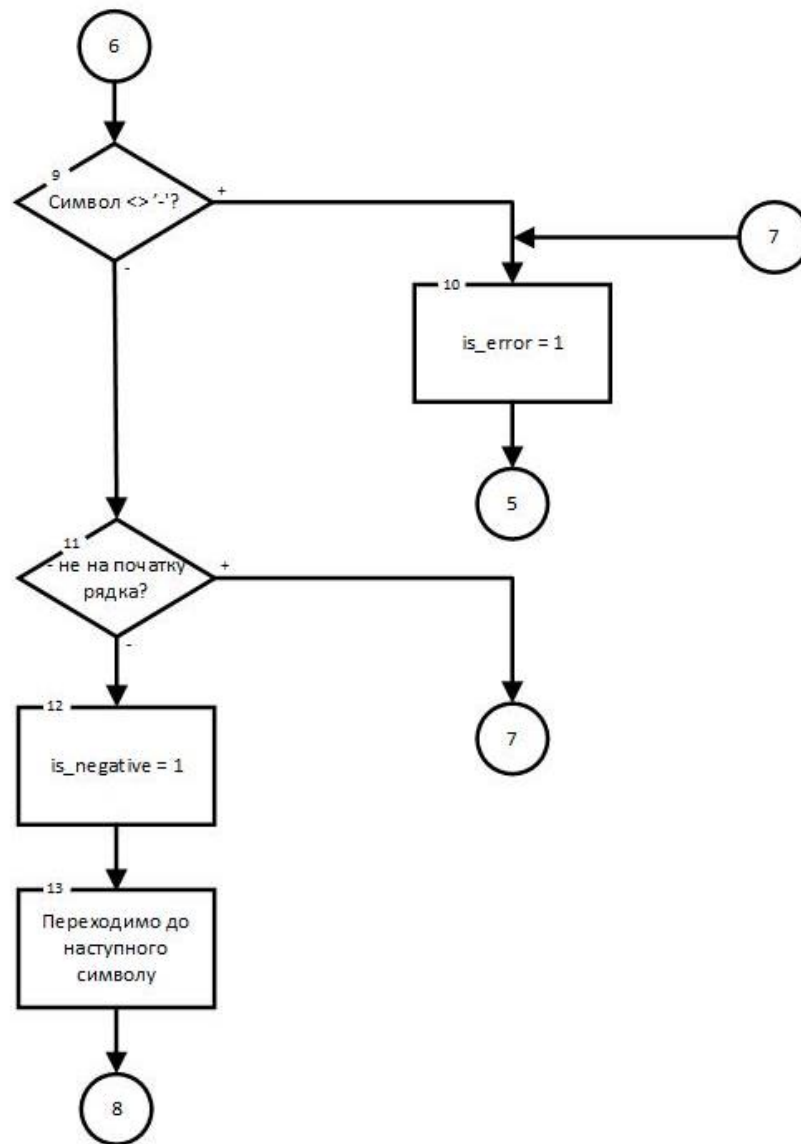


Рисунок 5. Схема процедури transform (продовження 1)

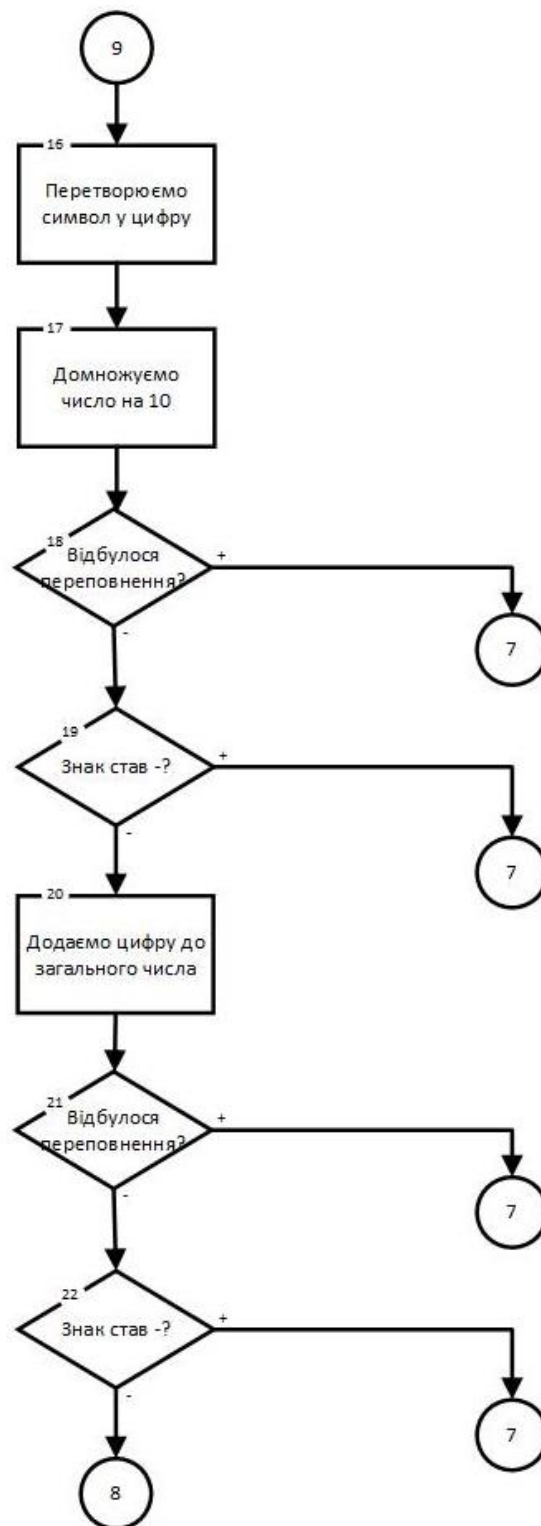


Рисунок 6. Схема процедури transform (продовження 2)

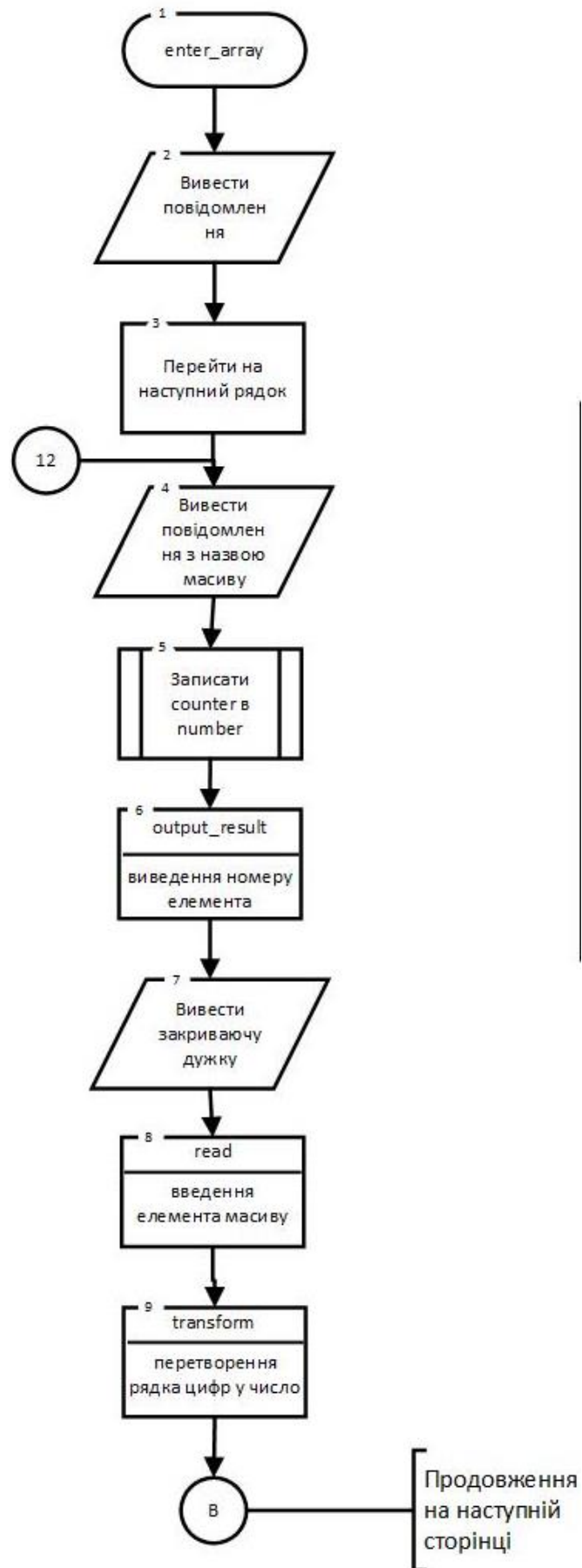


Рисунок 7. Схема процедури enter_array

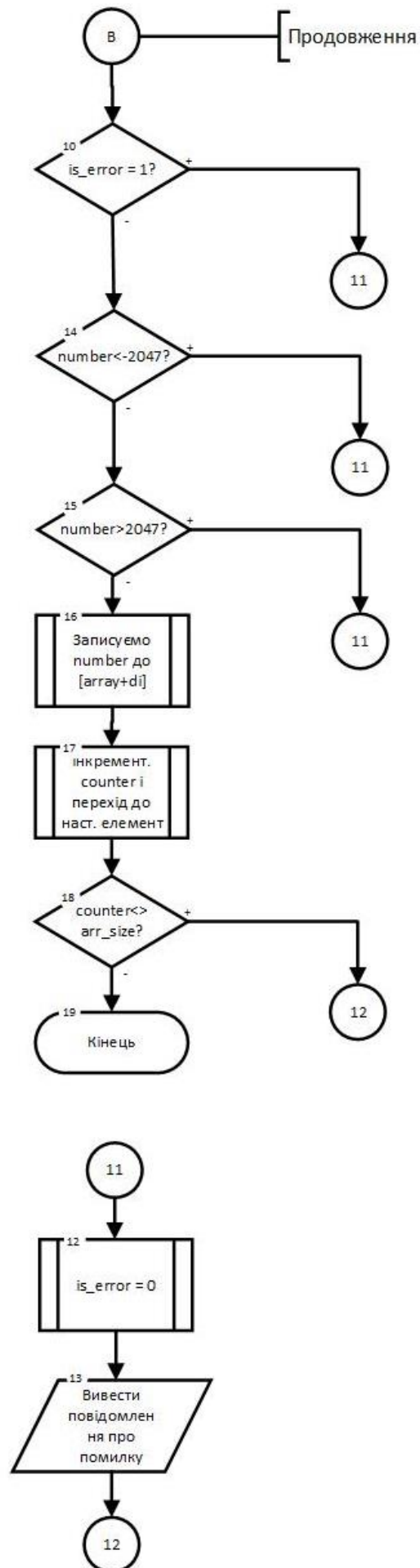
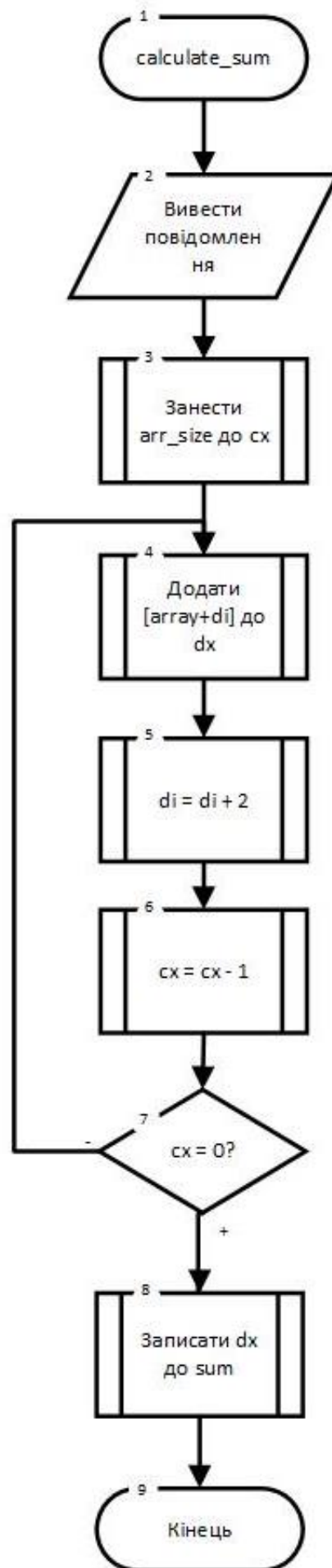
Рисунок 8. Схема процедури `enter_array` (продовження)



Рисунок 9. Схема процедури output_result

Рисунок 10. Схема процедури `calculate_sum`

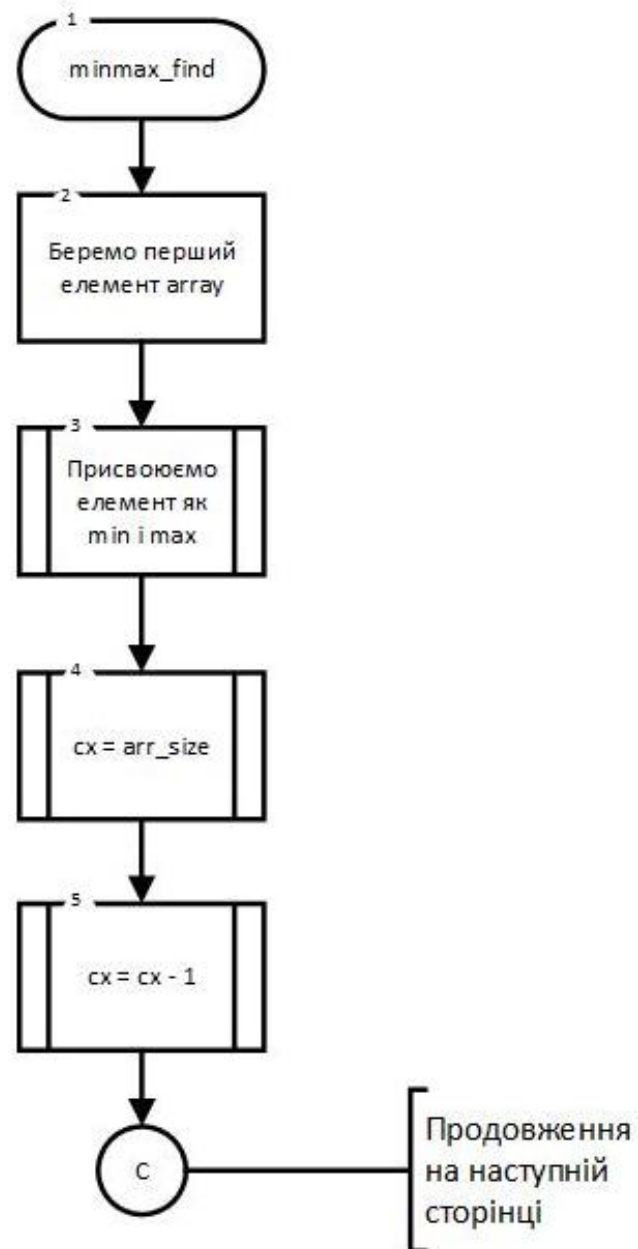
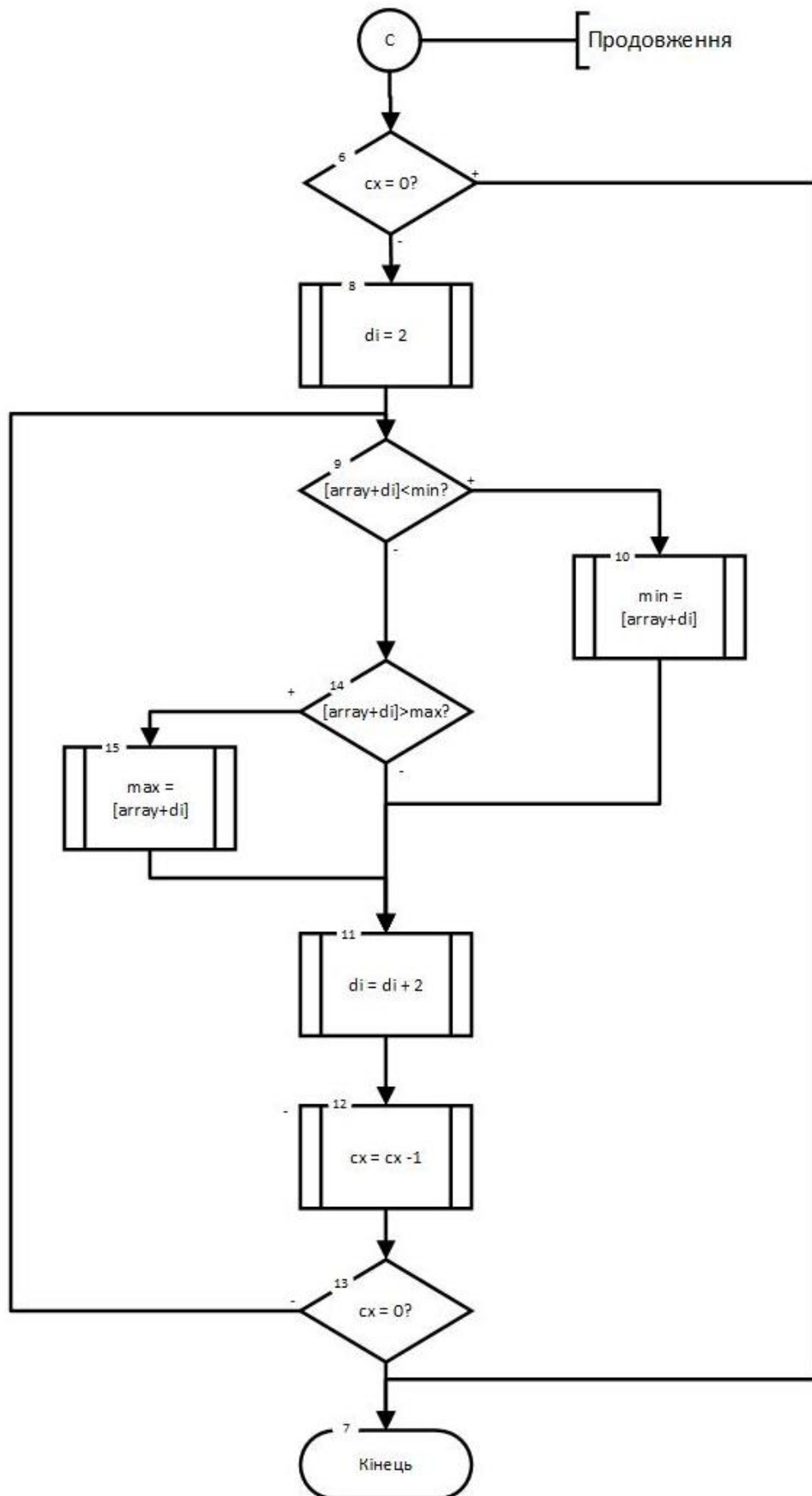


Рисунок 11. Схема процедури `minmax_find`

Рисунок 12. Схема процедури `minmax_find` (продовження)

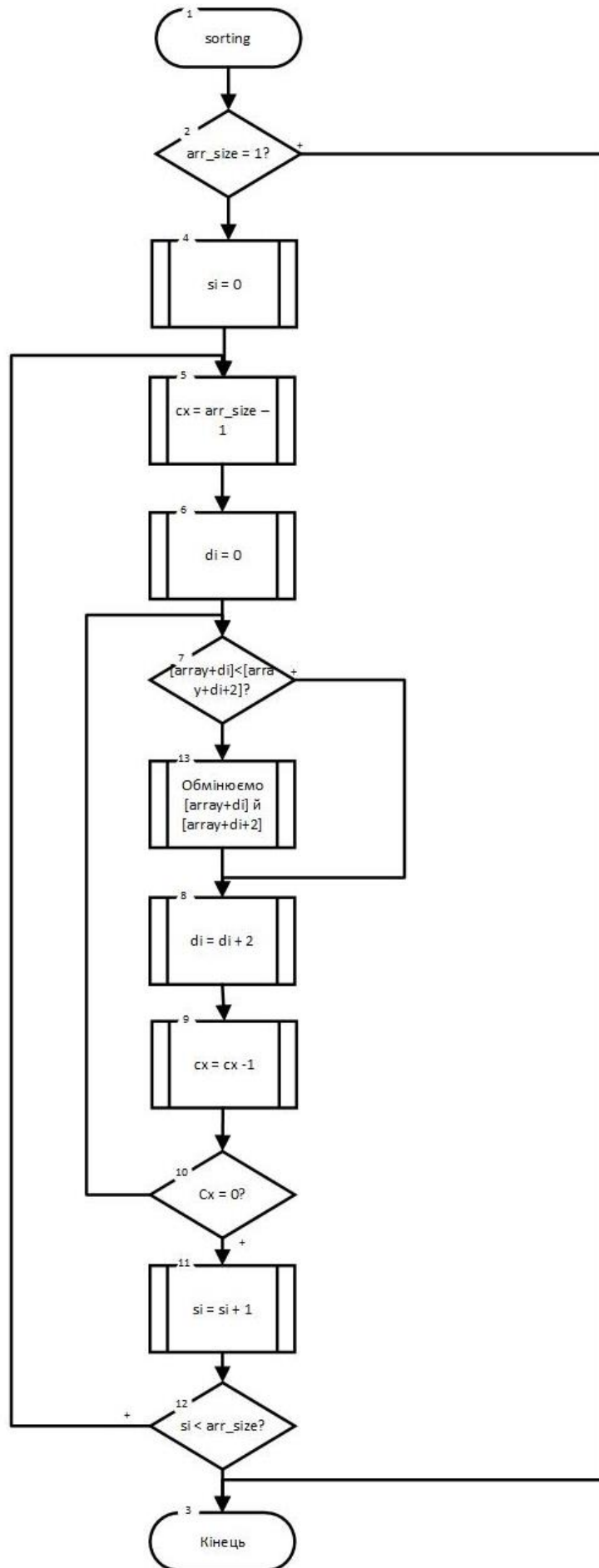


Рисунок 13. Схема процедури sorting

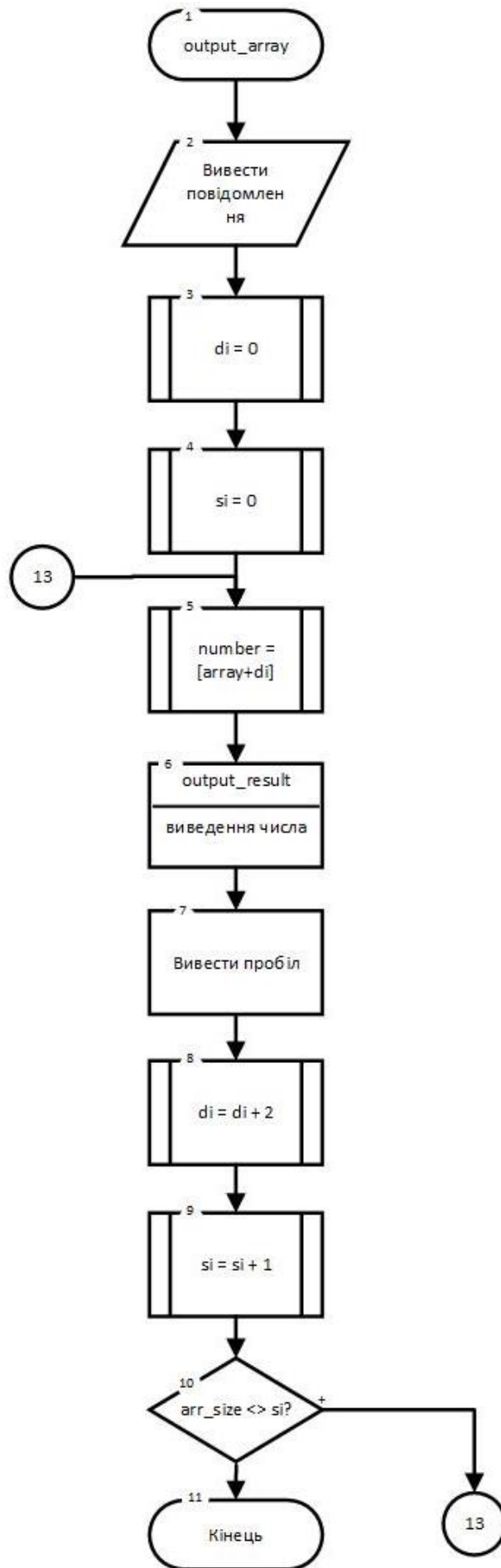


Рисунок 14. Схема процедури output_array

Програма 2.

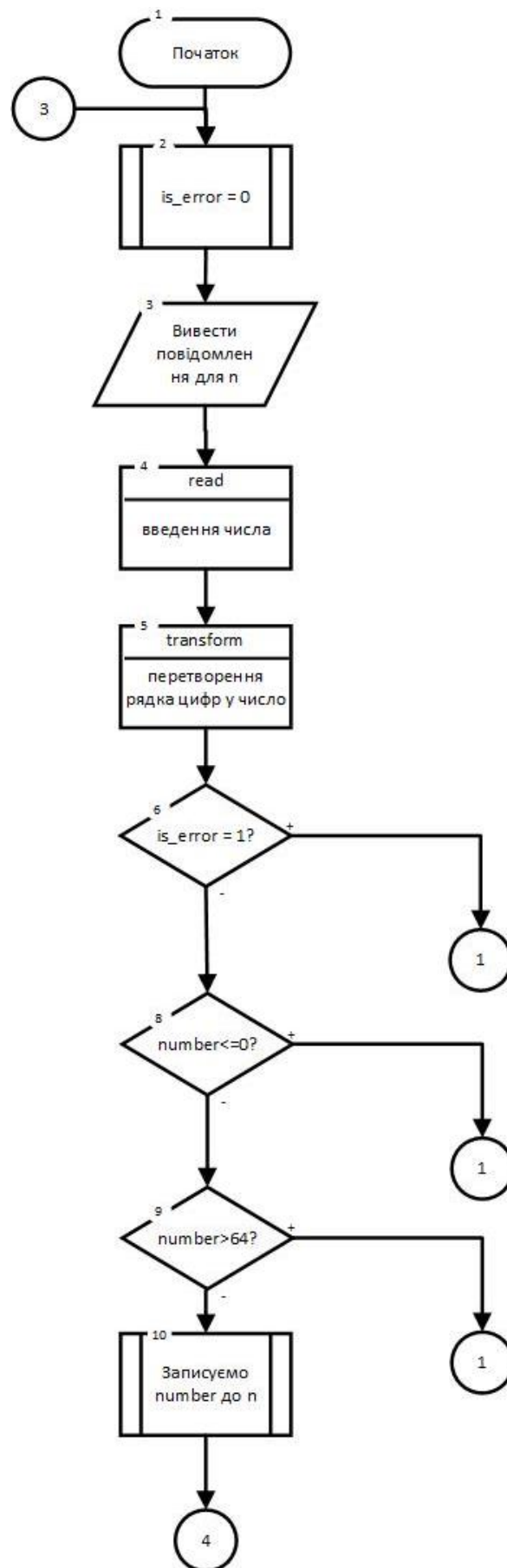


Рисунок 15. Схема основної частини програми

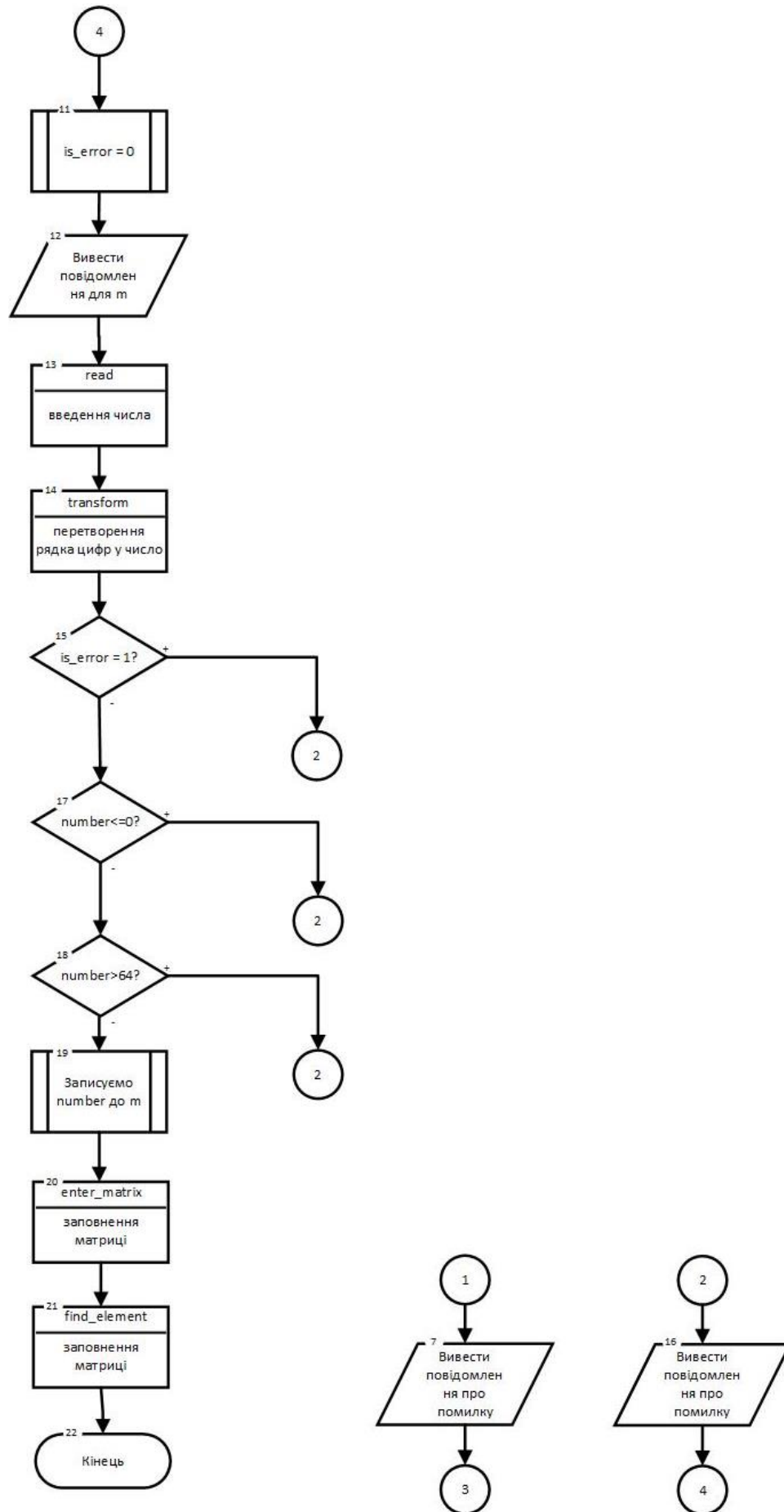


Рисунок 16. Схема основної частини програми (продовження)

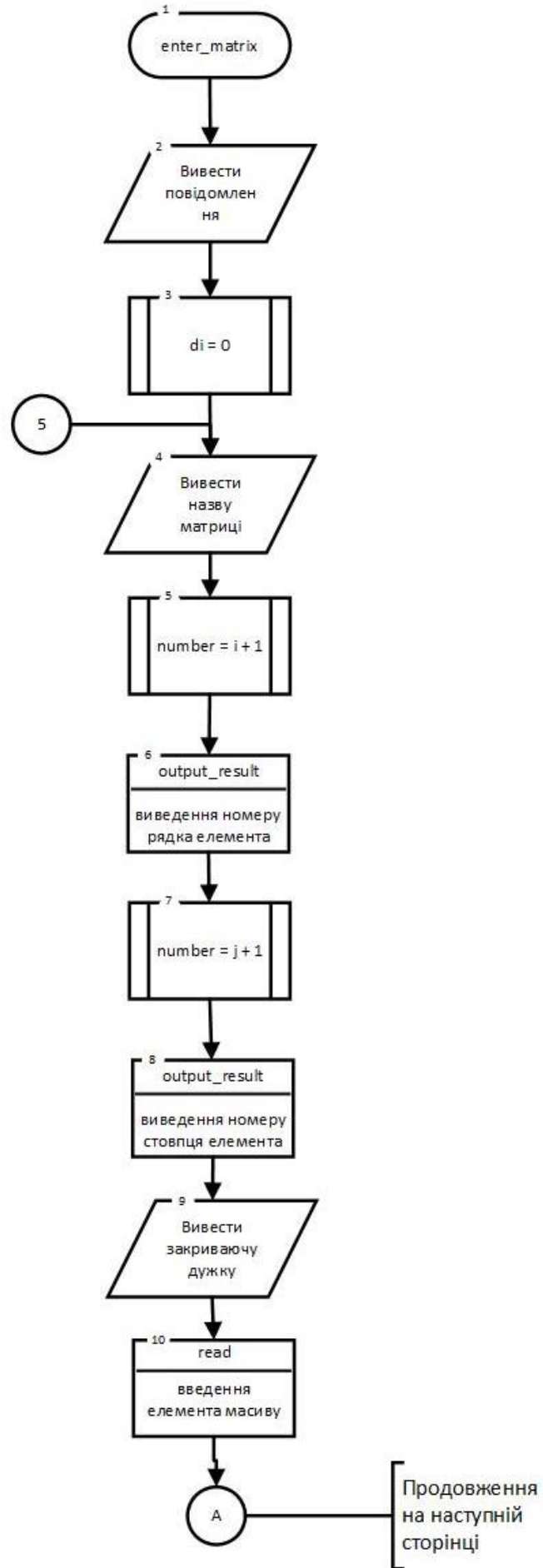
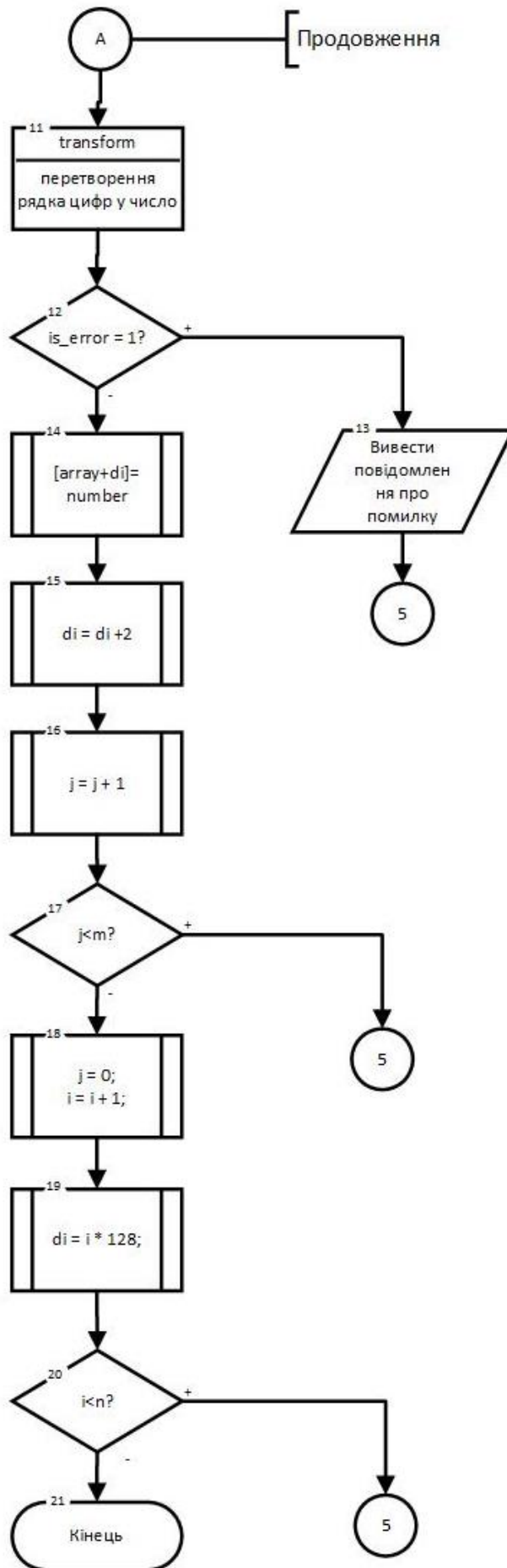


Рисунок 17. Схема процедури enter_matrix

Рисунок 17. Схема процедури `enter_matrix` (продовження)

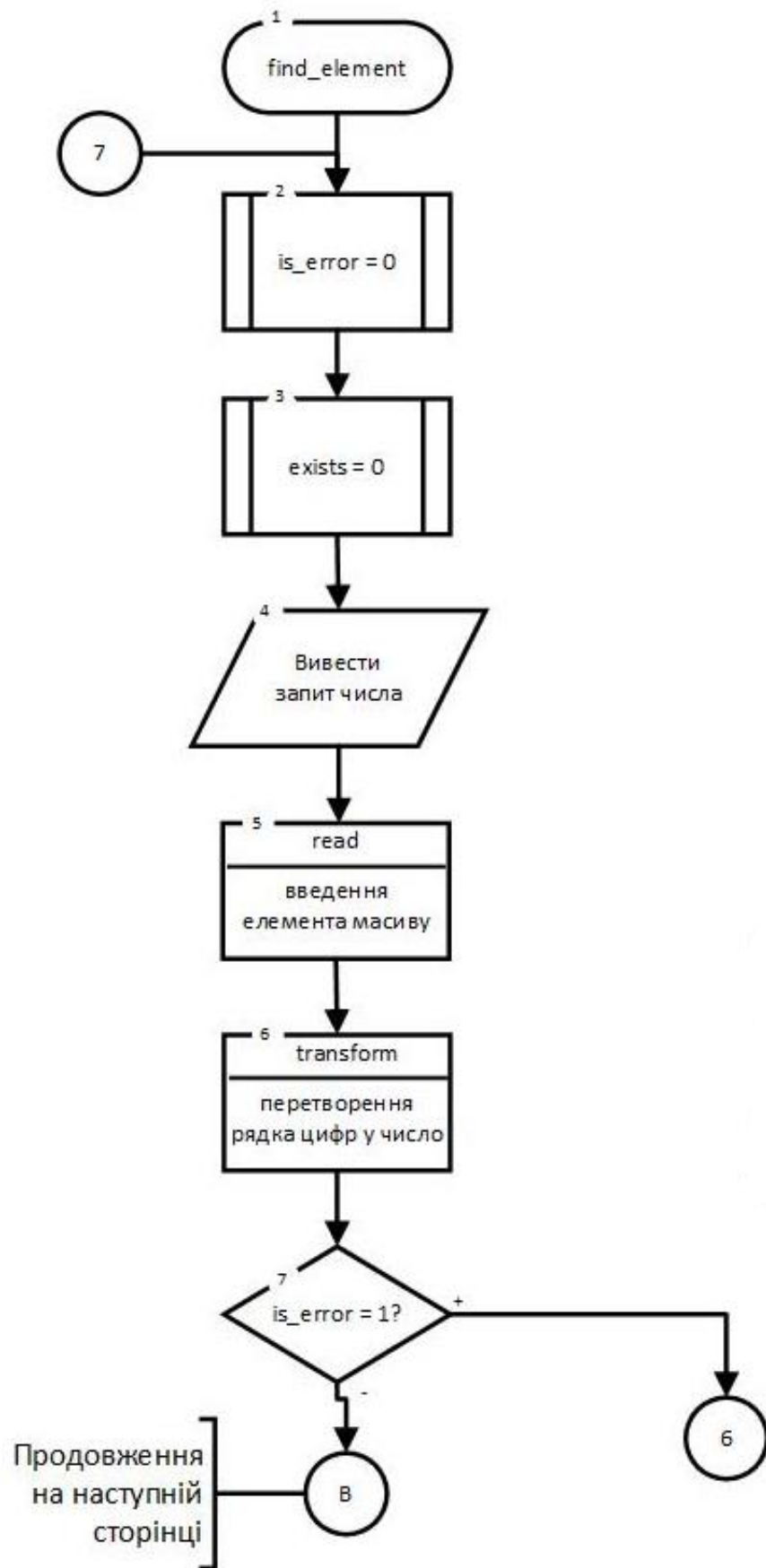
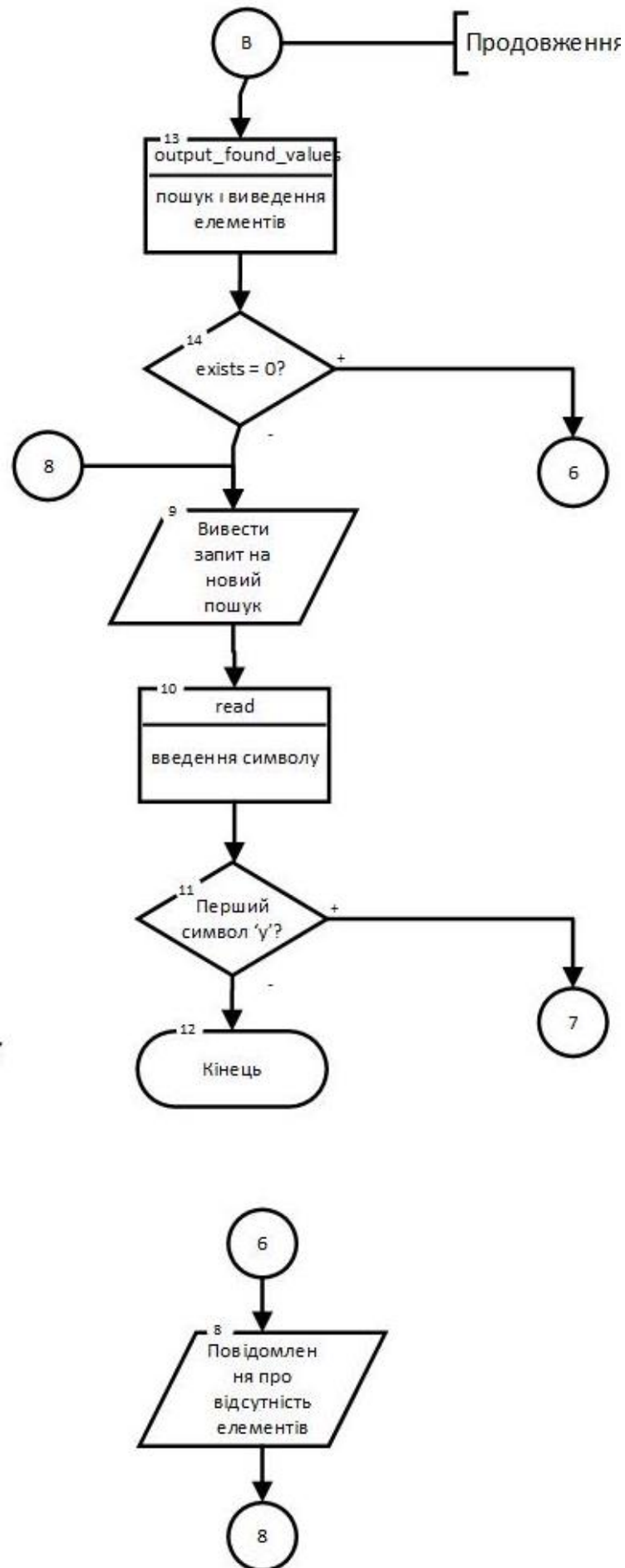


Рисунок 17. Схема процедури find_element

Рисунок 18. Схема процедури `find_element` (продовження)

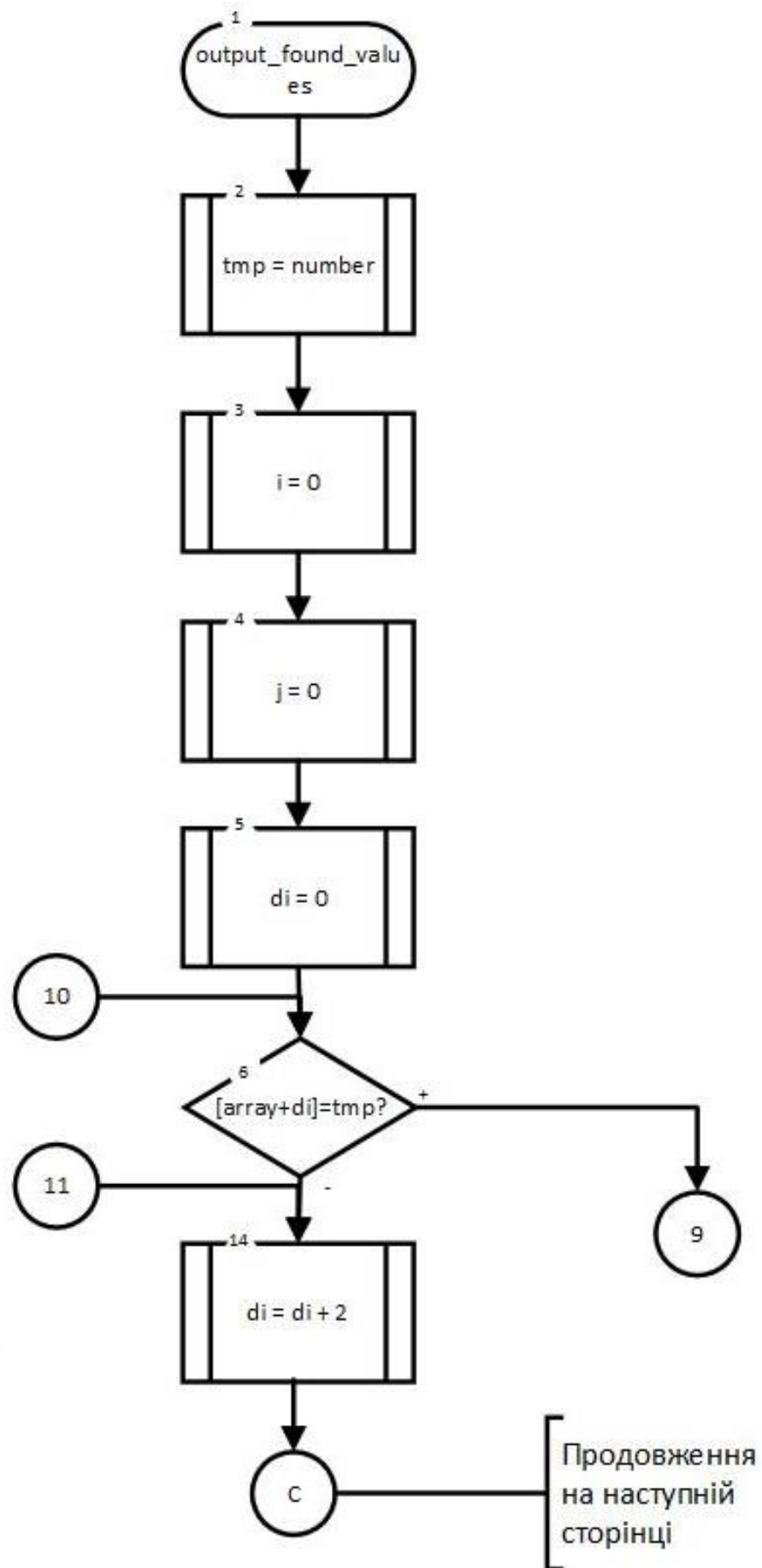


Рисунок 19. Схема процедури `output_found_values`

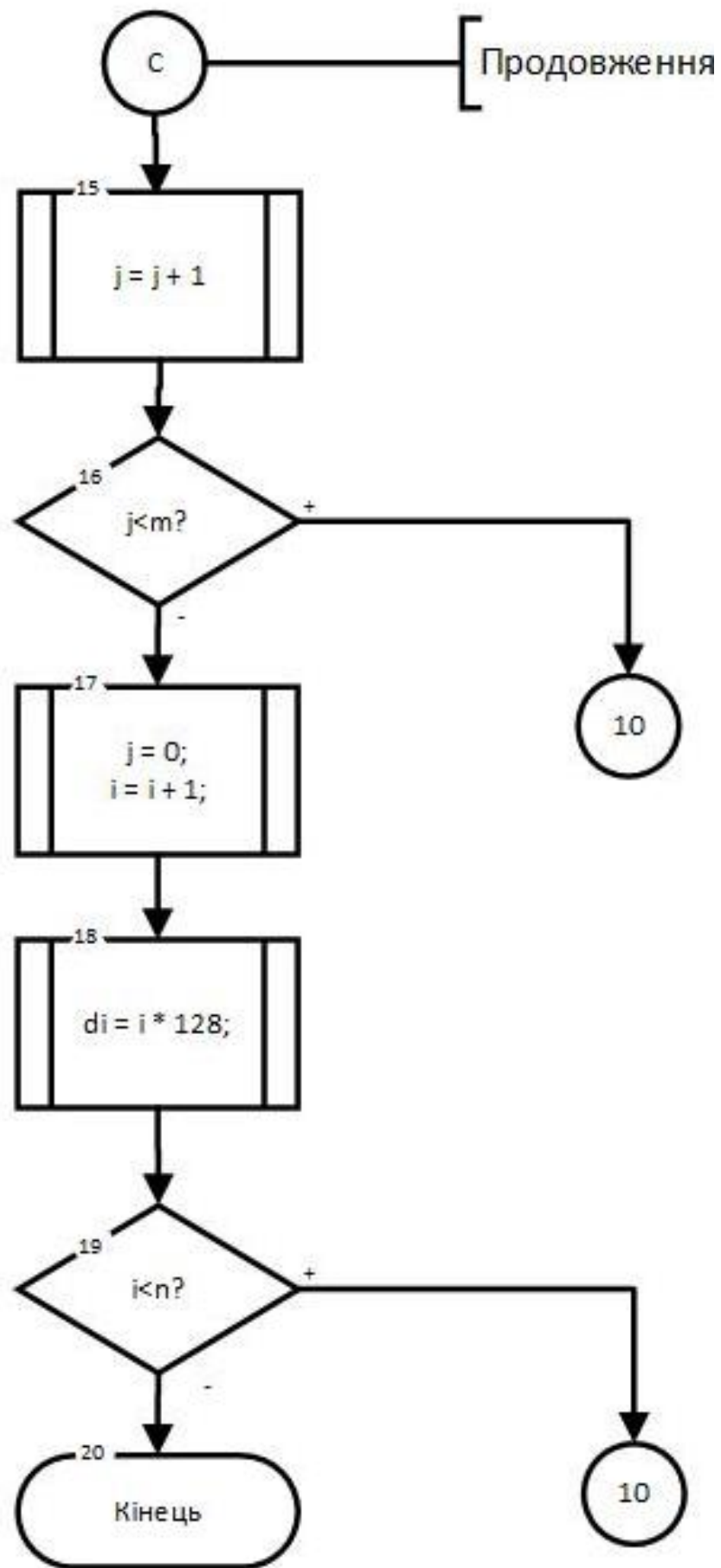


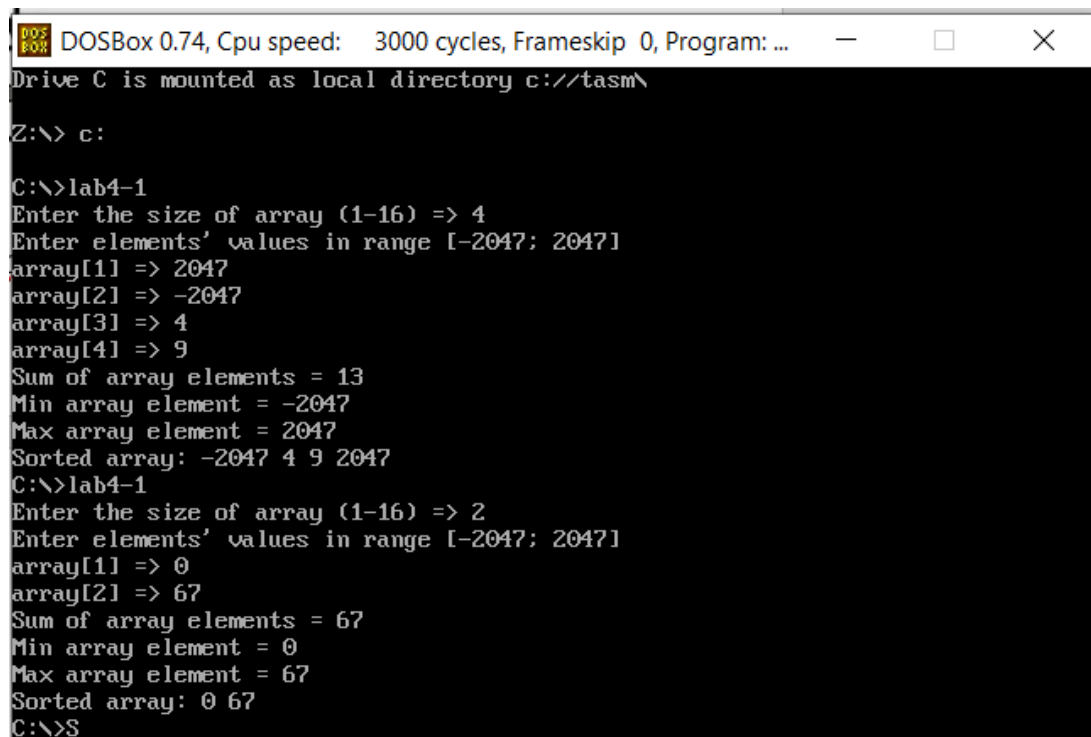
Рисунок 20. Схема процедури `output_found_values` (продовження 1)



Рисунок 21. Схема процедури `output_found_values` (продовження 2)

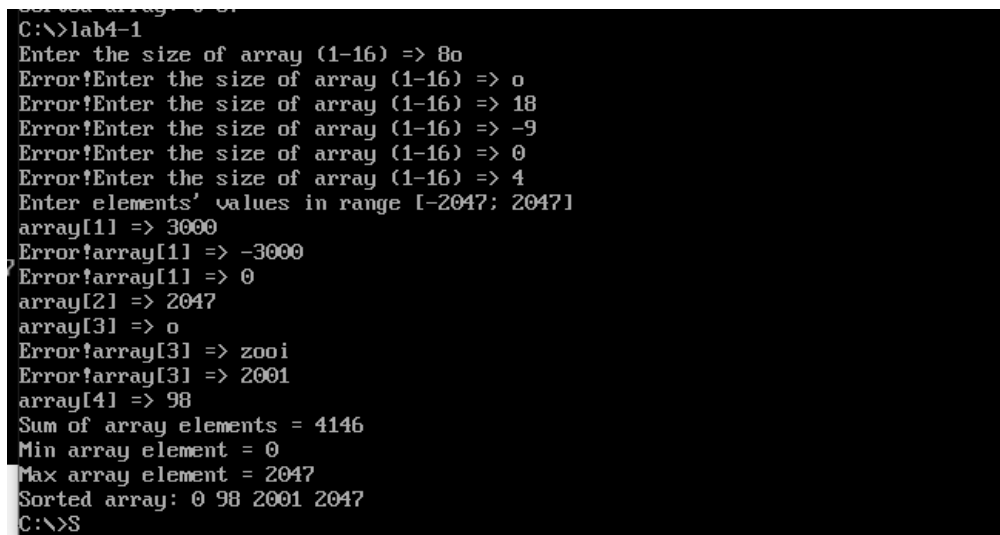
Приклад виконання програми.

Програма 1.



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Drive C is mounted as local directory c://tasm\
Z:\> c:
C:\> lab4-1
Enter the size of array (1-16) => 4
Enter elements' values in range [-2047; 2047]
array[1] => 2047
array[2] => -2047
array[3] => 4
array[4] => 9
Sum of array elements = 13
Min array element = -2047
Max array element = 2047
Sorted array: -2047 4 9 2047
C:\> lab4-1
Enter the size of array (1-16) => 2
Enter elements' values in range [-2047; 2047]
array[1] => 0
array[2] => 67
Sum of array elements = 67
Min array element = 0
Max array element = 67
Sorted array: 0 67
C:\> S
```

Рисунок 22. Приклад виконання програми з коректними даними.



```
C:\> lab4-1
Enter the size of array (1-16) => 80
Error!Enter the size of array (1-16) => 0
Error!Enter the size of array (1-16) => 18
Error!Enter the size of array (1-16) => -9
Error!Enter the size of array (1-16) => 0
Error!Enter the size of array (1-16) => 4
Enter elements' values in range [-2047; 2047]
array[1] => 3000
Error!array[1] => -3000
Error!array[1] => 0
array[2] => 2047
array[3] => 0
Error!array[3] => zooi
Error!array[3] => 2001
array[4] => 98
Sum of array elements = 4146
Min array element = 0
Max array element = 2047
Sorted array: 0 98 2001 2047
C:\> S
```

Рисунок 23. Приклад виконання програми з некоректними даними.

Програма 2.

```

C:\>lab4-2
Enter number of rows (1-64): 2
Enter number of columns (1-64): 4
Enter elements ([-32767; 32767])
matr[1][1] => 1234
matr[1][2] => 0
matr[1][3] => -4553
matr[1][4] => 1234
matr[2][1] => 1234
matr[2][2] => 56
matr[2][3] => 0
matr[2][4] => 0
What value do you want to find? : 0
matr[1][2]
matr[2][3]
matr[2][4]
Continue the search? ('y' for yes): y
What value do you want to find? : 1234
matr[1][1]
matr[1][4]
matr[2][1]
Continue the search? ('y' for yes): n
C:\>S

```

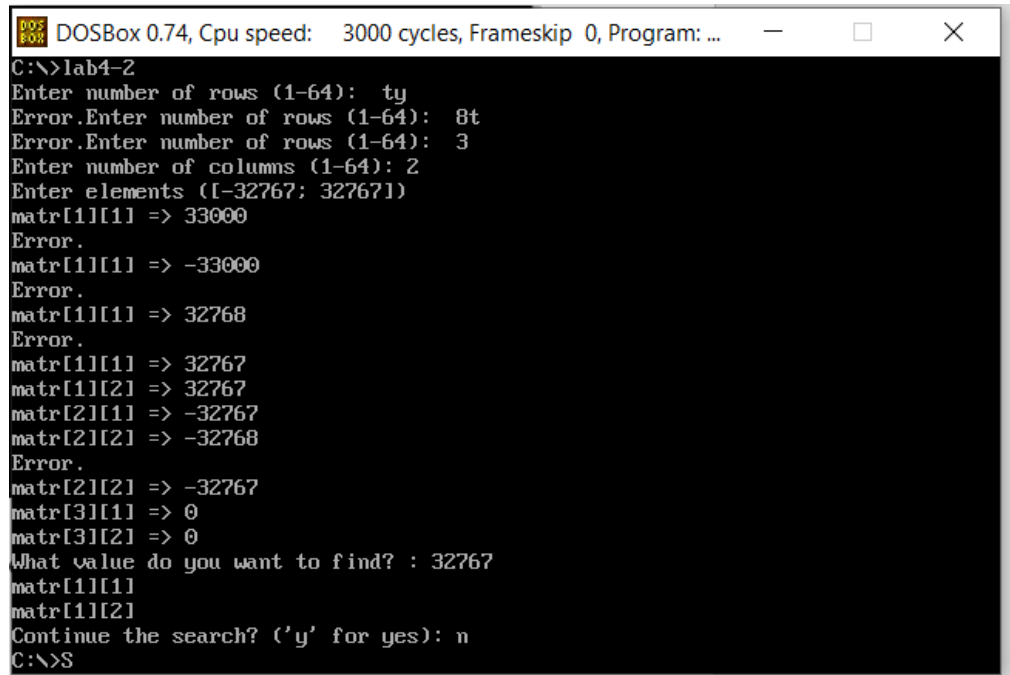
Рисунок 24. Приклад виконання програми з коректними даними.

```

C:\>lab4-2
Enter number of rows (1-64): 2
Enter number of columns (1-64): 2
Enter elements ([-32767; 32767])
matr[1][1] => 78
matr[1][2] => -900
matr[2][1] => 12
matr[2][2] => 54
What value do you want to find? : 0
Element is not found.
Continue the search? ('y' for yes): y
What value do you want to find? : -78
Element is not found.
Continue the search? ('y' for yes): y
What value do you want to find? : 78
matr[1][1]
Continue the search? ('y' for yes): n
C:\>S_

```

Рисунок 25. Приклад виконання програми, коли елемент не знайдено.



```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
C:\>lab4-2
Enter number of rows (1-64): ty
Error.Enter number of rows (1-64): 8t
Error.Enter number of rows (1-64): 3
Enter number of columns (1-64): 2
Enter elements ([-32767; 32767])
matr[1][1] => 33000
Error.
matr[1][1] => -33000
Error.
matr[1][1] => 32768
Error.
matr[1][1] => 32767
matr[1][2] => 32767
matr[2][1] => -32767
matr[2][2] => -32768
Error.
matr[2][2] => -32767
matr[3][1] => 0
matr[3][2] => 0
What value do you want to find? : 32767
matr[1][1]
matr[1][2]
Continue the search? ('y' for yes): n
C:\>S

```

Рисунок 26. Приклад виконання програми з некоректними даними.

Програма повідомляє про некоректне введення, якщо введено символи або ж значення, що перевищує задані межі. Якщо значення некоректне, програма виводить повідомлення й пропонує ввести значення ще раз. Після введення коректного значення програма виводить повідомлення про результат виконання.

Перевірні дані – розрахунки для програми 1.

<i>Вхідні дані</i>	<i>Розрахунки</i>	<i>Вихідні дані</i>
<i>Програма 1</i>		
arr_size = 4 array = [2047, -2047, 4, 9]	$sum = 2047 - 2047 + 4 + 9 = 13;$ $min(array) = -2047;$ $max(array) = 2047$	Sum = 13, min = -2047, max = 2047
arr_size = 2 array = [0, 67]	$sum = 0 + 67 = 67;$ $min(array) = 0;$ $max(array) = 67$	Sum = 67, min = 0, max = 67
arr_size = 4 array = [0, 2047, 2001, 98]	$sum = 0 + 2047 + 2001 + 98$ $= 4146;$ $min(array) = 0;$ $max(array) = 2047$	Sum = 4146, min = 0, max = 2047

Висновок.

Отже, у даній роботі я ознайомився із роботою з масивою.

Було написано програму, яка має наступний функціонал:

1. Програма з одномірним масивом:

- Можливість введення користувачем розміру одномірного масиву.
- Можливість введення користувачем значень елементів одномірного масиву.
- Можливість знаходження суми елементів одномірного масиву.
- Можливість пошуку максимального і мінімального елементів одномірного масиву.
- Можливість сортування одномірного масиву цілих чисел загального вигляду.

2. Програма з двомірним масивом:

- a. Можливість введення користувачем розміру двовірного масиву.
- b. Можливість введення користувачем значень елементів двовірного масиву.
- c. Можливість пошуку координат всіх входжень заданого елемента в двовірному масиві, елементи масиву та пошуковий елемент вводять користувач.

Програми мають захист від некоректного введення вхідних даних (символи, переповнення). Програми було скопійовано, налагоджено та виконано. У результаті виконання програм отримується коректний й очікуваний результат, відповідно до попередніх розрахунків.