НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт до комп’ютернего практикуму з дисципліни

«Системне програмне забезпечення»

Прийняв
асистент кафедри ІПІ
Пархоменко А.В.
“21” травня 2023 р.

Виконав
Студент групи ІП-15
Мєшков А. І.

Київ – 2023

Комп‘ютерний практикум № 4

Масиви

Загальні положення

Викладені в лекційному матеріалі.

Завдання комп’ютерного практикуму №4

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

Текст програми

```
STSEG SEGMENT PARA STACK "STACK"
   dw 64 DUP ( '?' )
STSEG ENDS

DSEG SEGMENT PARA PUBLIC "DATA"

   ;commandRet dw 0
   progNumber dw 0
   readNumber dw 0
   fl dw 0
   errorFl dw 0
   len dw 0
   numstr db 7,?,7 dup('?')
   error_mes db "Error$"
   command_mes db "Enter number of program -> $"
   countOdn dw 0
   countOdn_mes db "Enter count -> $"
   odnMas dw 100 dup (0)
   elemMas_mes db "Enter element number $"
   arrow db " -> $"
   space db " $"
   i dw 0
   j dw 0
   writeNumber dw 0
   memoryCx dw 0
   memoryCx2 dw 0
   sum dw 0
   sum_mes db "Sum: $"
   max dw 0
   min dw 0
   max_mes db "Max: $"
   min_mes db "Min: $"
   firstElem dw 0
   secondElem dw 0
   mtx dw 10000 dup(0)
   countStr dw 0
   countStlb dw 0
   str_mes db "Enter heigh -> $"
   stlb_mes db "Enter weigh -> $"
   step dw 0
   findElem dw 0
   find_mes db "Enter searched element -> $"
   isFind dw 0
   notFind_mes db "not find$"

DSEG ENDS
```

```
CSEG SEGMENT PARA PUBLIC "CODE"
ASSUME ds:DSEG, cs:CSEG, SS:STSEG

MAIN PROC FAR

    PUSH ds
    MOV ax, 0
    PUSH ax
    MOV ax,DSEG
    MOV ds,ax

    CALL READ_COMMAND
    CMP progNumber,1
    je pr1
    CMP progNumber,2
    je pr2
    CMP progNumber,3
    je pr3
    CMP progNumber,4
    je pr4
pr1:
  CALL PROG1
pr2:
  CALL PROG2
pr3:
  CALL PROG3
pr4:
  CALL PROG4
prog_end:
  MOV ah,4Ch
  int 21h

MAIN ENDP

ERROR PROC NEAR
  MOV errorFl,1
  MOV dx, offset error_mes
  MOV ah,9
  int 21h
  MOV al,10
  int 29h
  RET
ERROR ENDP

READ PROC NEAR

start:
```

```
  MOV readNumber,0
  MOV fl,0
  MOV len,0

  ;read number
  LEA dx,numstr
  MOV ah,10
  int 21h

  MOV al,10
  int 29h

  MOV ax,0
  MOV al,[numstr+1]
  MOV len,ax

  MOV si,2
  MOV al,numstr[si]

  ;translate to int
  CMP al,2Dh
  jne no_minus

  inc si
  MOV fl,1
  dec len

no_minus:
  MOV cx,len
  MOV bx,10

cycle:
  MOV ax,readNumber
  IMUL bx
  jo toError
  MOV readNumber,ax
  MOV ax,0
  MOV al,numstr[si]
  SUB al,30h
  CMP al,0
  jl toError
  CMP al,9
  ja toError
  ADD readNumber,ax
  CMP readNumber,32767
  ja toError
  inc si
  loop cycle
```

```
  CMP fl,1
  jne finish
  CMP readNumber,32767
  ja toError
  NEG readNumber
  jmp finish

toError:
  CALL ERROR
finish:
  RET
READ ENDP

READ_COMMAND PROC NEAR

point1:
  MOV dx,offset command_mes
  MOV ah,9
  int 21h
  MOV errorFl,0
  CALL READ
  CMP errorFl,1
  je point1
  CMP readNumber,1
  jl toError1
  CMP readNumber,4
  ja toError1
  MOV ax,readNumber
  MOV progNumber,ax
  RET
toError1:
  CALL ERROR
  jmp point1
  RET

READ_COMMAND ENDP

WRITE PROC NEAR

  MOV bx,writeNumber
  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

WRITE ENDP

READ_ODN_MAS PROC NEAR

point2:
  MOV dx,offset countOdn_mes
  MOV ah,9
  int 21h
  MOV errorFl,0
  CALL READ
  CMP errorFl,1
  je point2
  CMP readNumber,1
  jl toError2
  CMP readNumber,100
  ja toError2
  MOV ax,readNumber
  MOV countOdn,ax
  MOV cx,countOdn
  MOV i,1
  MOV si,0
  LEA di,odnMas
readMas:
    jmp d
point3:
    MOV cx,memoryCx
d:
    MOV dx,offset elemMas_mes
    MOV ah,9
    int 21h
    MOV ax,i
    MOV writeNumber,ax
```

```
    MOV memoryCx,cx
    CALL WRITE
    MOV dx,offset arrow
    MOV ah,9
    int 21h
    MOV errorFl,0
    CALL READ
    CMP errorFl,1
    je point3
    MOV cx,memoryCx
    MOV ax,readNumber
    MOV [di],ax
    ADD i,1
    ADD di,2
loop readMas
    RET
toError2:
    CALL ERROR
    jmp point2
    RET

READ_ODN_MAS ENDP

WRITE_ODN_MAS PROC NEAR

    MOV cx,countOdn
    MOV si,0
write_mas:
    MOV dx,odnMas[si]
    MOV writeNumber,dx
    MOV memoryCx,cx
    CALL WRITE
    MOV cx, memoryCx
    MOV dx,offset space
    MOV ah,9
    int 21h
    ADD si,2
loop write_mas
    RET

WRITE_ODN_MAS ENDP

PROG1 PROC NEAR

    CALL READ_ODN_MAS
    CALL WRITE_ODN_MAS
    MOV al,10
    int 29h
```

```
    MOV cx,countOdn
    MOV sum,0
    MOV si,0
summary:
    MOV ax,odnMas[si]
    ADD sum,ax
    jo toError3
    ADD si,2
loop summary
    MOV dx,offset sum_mes
    MOV ah,9
    int 21h
    MOV ax,sum
    MOV writeNumber,ax
    CALL WRITE
    jmp prog_end
    RET
toError3:
    CALL ERROR
    jmp prog_end

PROG1 ENDP

PROG2 PROC NEAR

    CALL READ_ODN_MAS
    CALL WRITE_ODN_MAS
    MOV al,10
    int 29h
    MOV cx,countOdn
    MOV si,0
    MOV ax,odnMas[si]
    MOV max,ax
    MOV min,ax
    ADD si,2
    DEC cx
max_find:
    MOV ax,odnMas[si]
    CMP ax,max
    jg find_max
    CMP ax,min
    jl find_min
    jmp con
find_max:
    MOV max,ax
    jmp con
find_min:
    MOV min,ax
```

```
con:
   ADD si,2
loop max_find
   MOV dx,offset max_mes
   MOV ah,9
   int 21h
   MOV ax,max
   MOV writeNumber,ax
   CALL WRITE
   MOV al,10
   int 29h
   MOV dx,offset min_mes
   MOV ah,9
   int 21h
   MOV ax,min
   MOV writeNumber,ax
   CALL WRITE
   jmp prog_end
   RET

PROG2 ENDP

PROG3 PROC NEAR
   CALL READ_ODN_MAS
   CALL WRITE_ODN_MAS
   MOV al,10
   int 29h
   MOV cx,countOdn
first_cycle:
   MOV memoryCx,cx
   MOV cx, countOdn
   DEC cx
   LEA di,odnMas
   MOV si,0
second_cycle:
   MOV ax,odnMas[si]
   MOV firstElem,ax
   ADD si,2
   MOV ax,odnMas[si]
   MOV secondElem,ax
   CMP firstElem,ax
   jle cycle_end
   MOV [di],ax
   ADD di,2
   MOV ax,firstElem
   MOV [di],ax
   jmp cycle_end_end
cycle_end:
```

```
    ADD di,2
cycle_end_end:
loop second_cycle
    MOV cx,memoryCx
loop first_cycle
    CALL WRITE_ODN_MAS
    jmp prog_end
    RET
PROG3 ENDP

READ_MTX PROC NEAR

    jmp point5
to_error5:
    CALL ERROR
to_error6:
    CALL ERROR
    jmp point6
point5:
    MOV dx,offset str_mes
    MOV ah,9
    int 21h
    MOV errorFl,0
    CALL READ
    CMP errorFl,1
    je point5
    CMP readNumber,1
    jl to_error5
    CMP readNumber,100
    jg to_error5
    MOV ax,readNumber
    MOV countStr,ax
point6:
    MOV dx,offset stlb_mes
    MOV ah,9
    int 21h
    MOV errorFl,0
    CALL READ
    CMP errorFl,1
    je point6
    CMP readNumber,1
    jl to_error6
    CMP readNumber,100
    jg to_error6
    MOV ax,readNumber
    MOV countStlb,ax
    MOV i,1
    MOV j,1
```

```
    MOV cx,countStr
    LEA di,mtx
cycle_str:
    MOV memoryCx,cx
    MOV j,1
    MOV cx,countStlb
cycle_stlb:
    jmp point7dop
point7:
    inc cx
point7dop:
    MOV memoryCx2,cx
    MOV dx,offset elemMas_mes
    MOV ah,9
    int 21h
    MOV ax,i
    MOV writeNumber,ax
    CALL WRITE
    MOV dx,offset space
    MOV ah,9
    int 21h
    MOV ax,j
    MOV writeNumber,ax
    CALL WRITE
    MOV dx,offset arrow
    MOV ah,9
    int 21h
    MOV errorFl,0
    CALL READ
    CMP errorFl,1
    je point7
    MOV ax,readNumber
    MOV [di],ax
    ADD di,2
    ADD j,1
    MOV cx,memoryCx2
loop cycle_stlb
    MOV cx,memoryCx
    ADD i,1
loop cycle_str
    RET

READ_MTX ENDP

WRITE_MTX PROC NEAR

    MOV cx,countStr
    MOV si,0
```

```
write_cycle1:
   MOV memoryCx,cx
   MOV cx,countStlb
write_cycle2:
   MOV memoryCx2,cx
   MOV ax,mtx[si]
   MOV writeNumber,ax
   CALL WRITE
   MOV dx,offset space
   MOV ah,9
   int 21h
   ADD si,2
   MOV cx,memoryCx2
loop write_cycle2
   MOV al,10
   int 29h
   MOV cx,memoryCx
loop write_cycle1
   RET

WRITE_MTX ENDP

PROG4 PROC NEAR
   CALL READ_MTX
   CALL WRITE_MTX
point8:
   MOV dx,offset find_mes
   MOV ah,9
   int 21h
   MOV errorFl,0
   CALL READ
   CMP errorFl,1
   je point8
   MOV ax,readNumber
   MOV findElem,ax
   MOV si,0
   MOV i,1
   MOV cx,countStr
find_cycle1:
   MOV memoryCx,cx
   MOV j,1
   MOV cx,countStlb
find_cycle2:
   MOV memoryCx2,cx
   MOV ax,mtx[si]
   CMP ax,findElem
   jne point9
   MOV ax,i
```

```
    MOV writeNumber,ax
    CALL WRITE
    MOV dx,offset space
    MOV ah,9
    int 21h
    MOV ax,j
    MOV writeNumber,ax
    CALL WRITE
    MOV al,10
    int 29h
    MOV isFind,1
point9:
    ADD j,1
    ADD si,2
    MOV cx,memoryCX2
loop find_cycle2
    ADD i,1
    MOV cx,memoryCx
loop find_cycle1
    MOV al,10
    int 29h
    CMP isFind,1
    je point10
    MOV dx,offset notFind_mes
    MOV ah,9
    int 21h
point10:
    jmp prog_end
    RET
PROG4 ENDP

CSEG ENDS

END MAIN
```

Схема функціонування програми

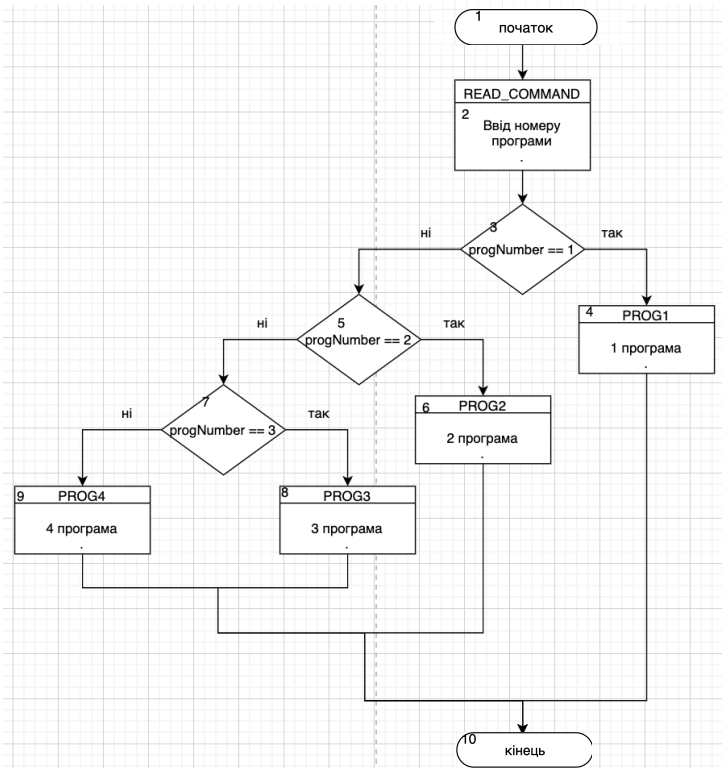


Рисунок 4.1 Схема функціонування головної процедури

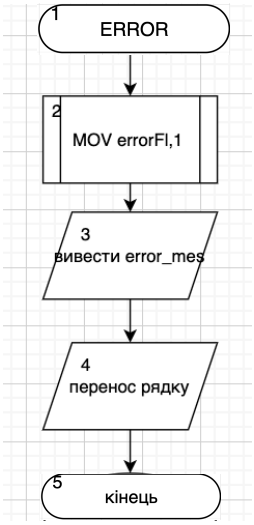


Рисунок 4.2 Схема функціонування програми виведення помилки

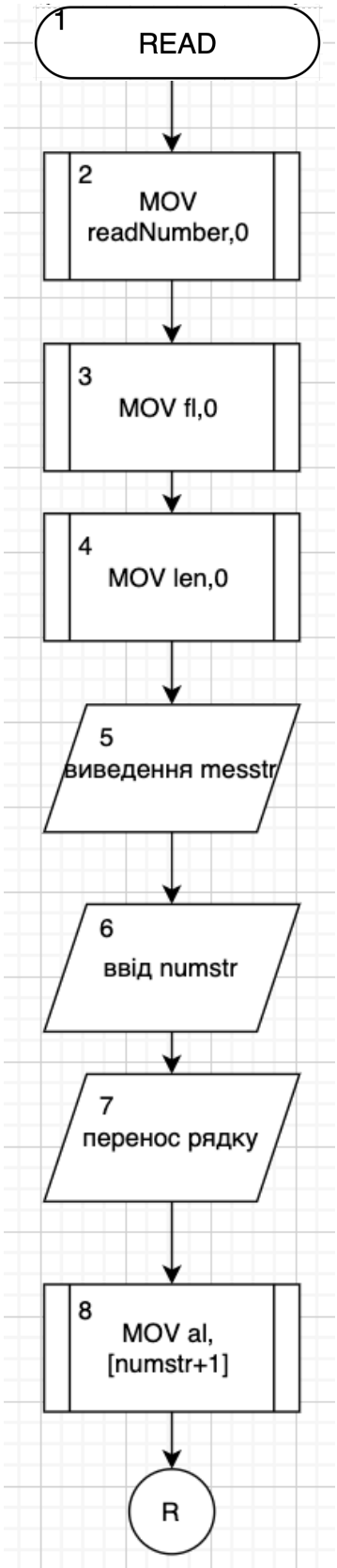
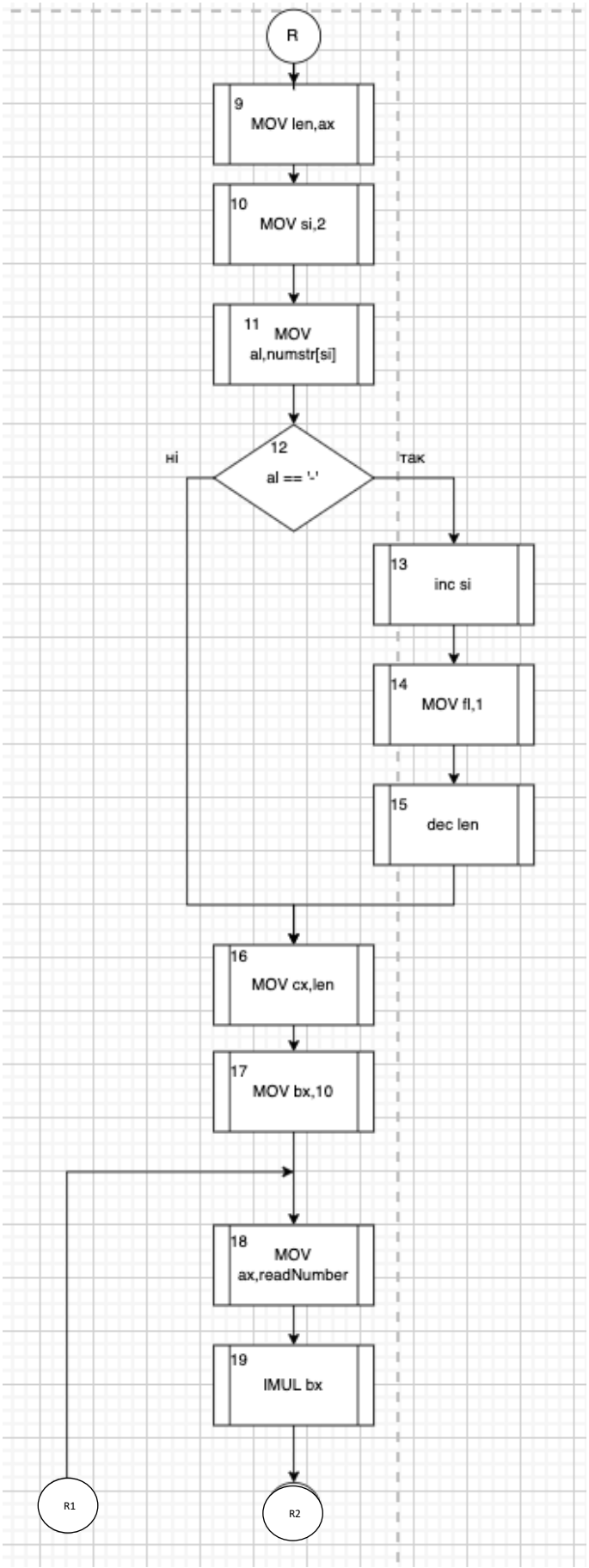
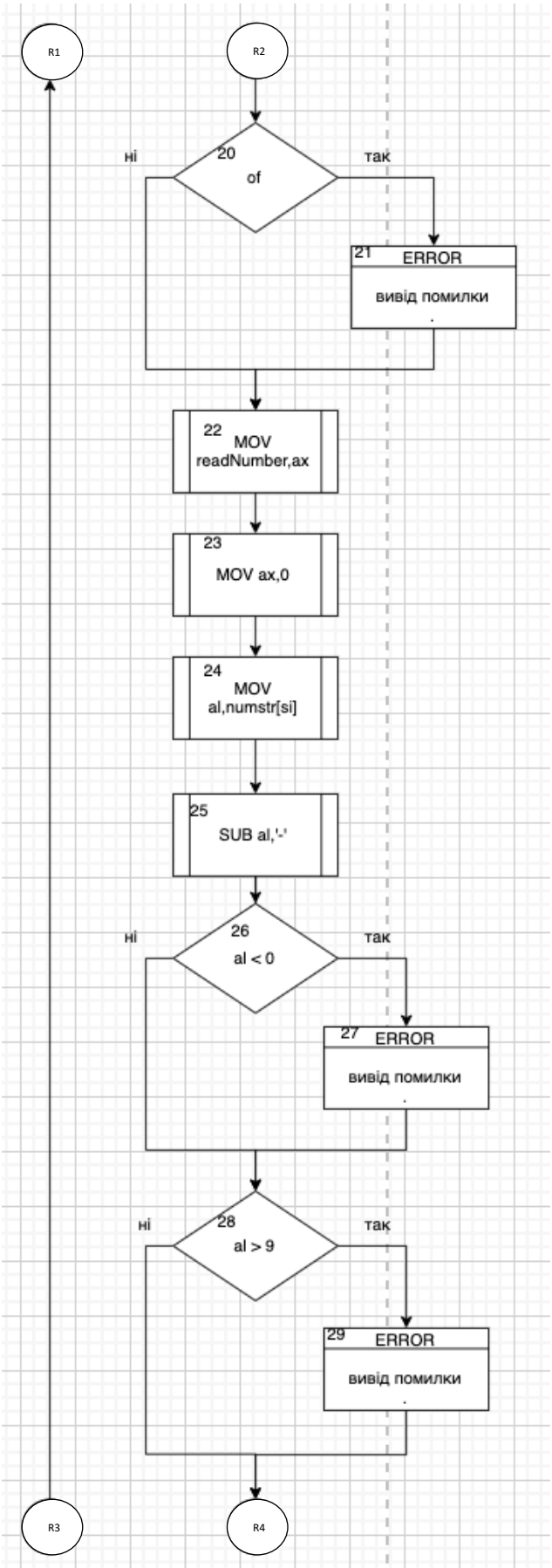


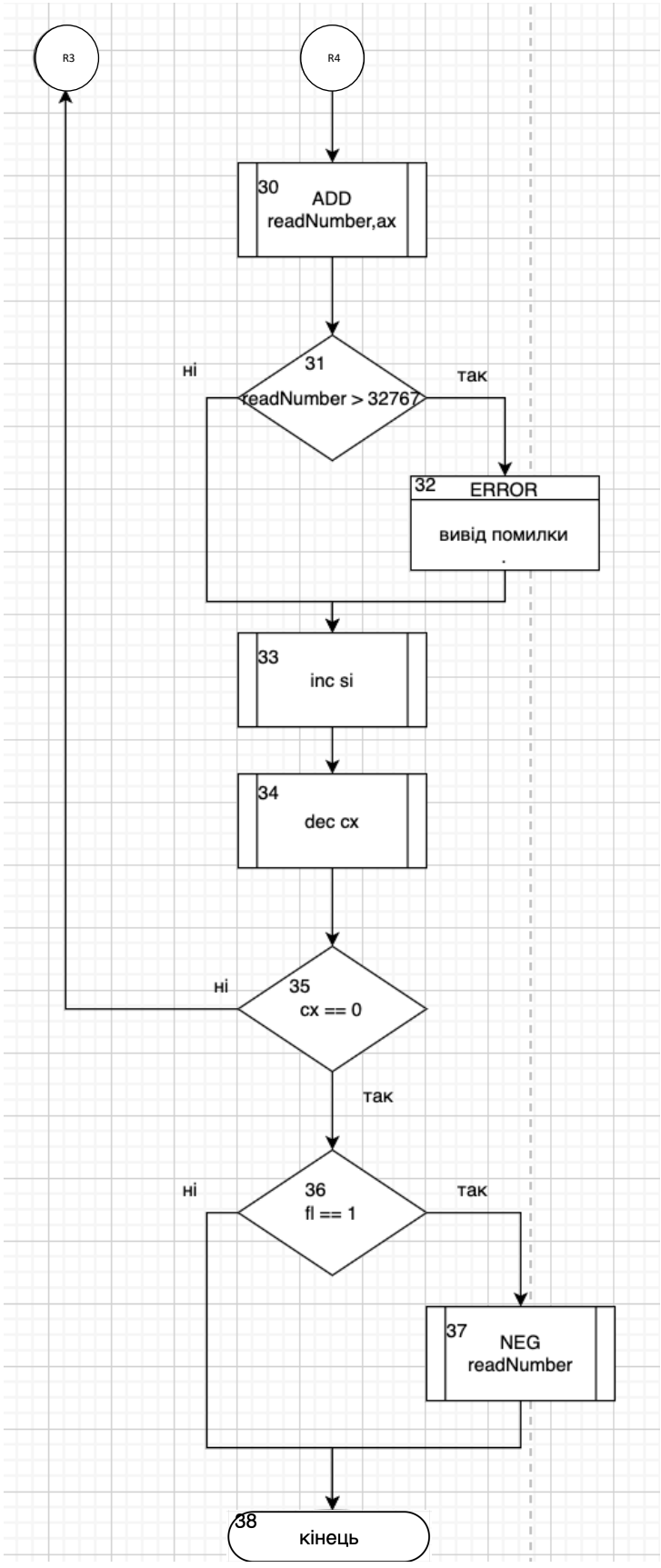
Рисунок 4.3 Схема функціонування процедури читання числа



Продовження рисунку 4.3



Продовження рисунку 4.3



Продовження рисунку 4.3

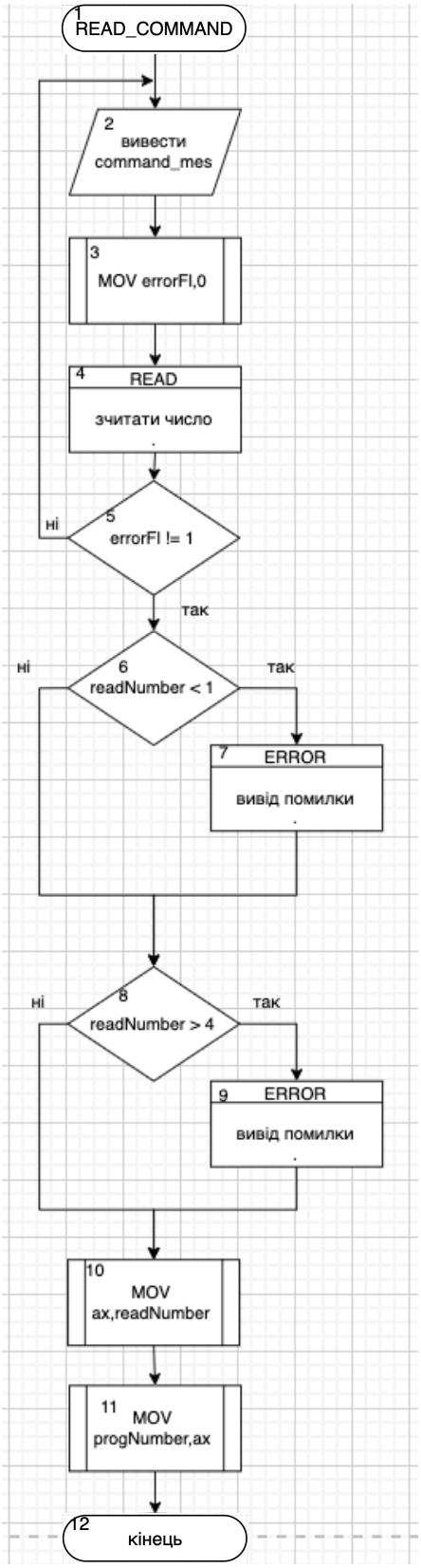


Рисунок 4.4 Схема функціонування процедури читання номеру програми

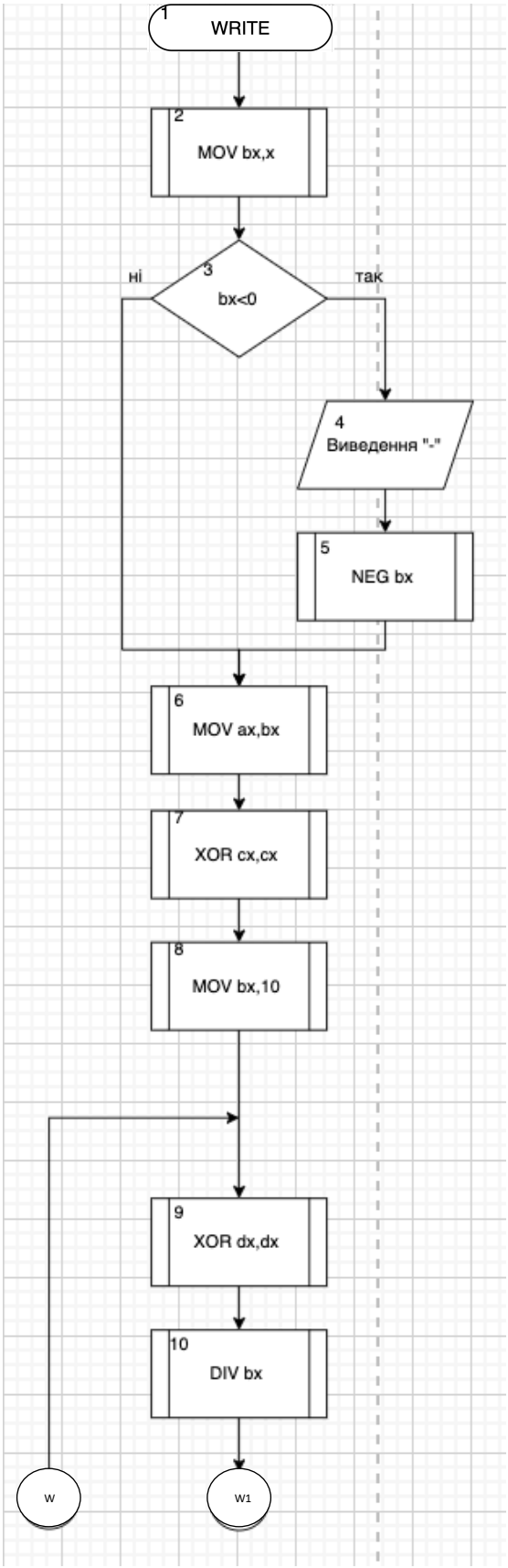
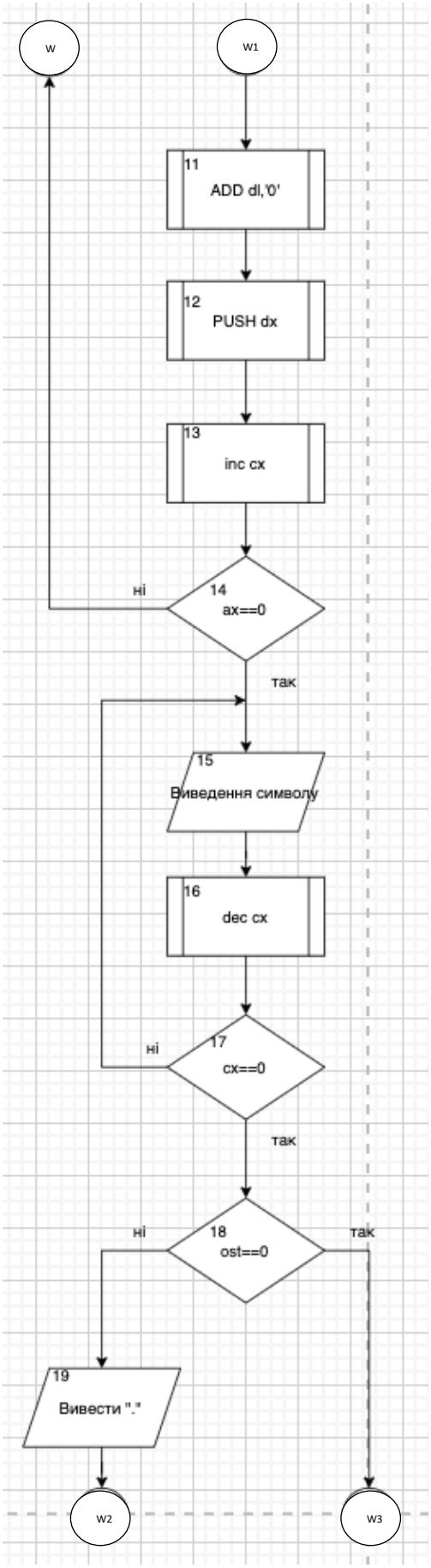
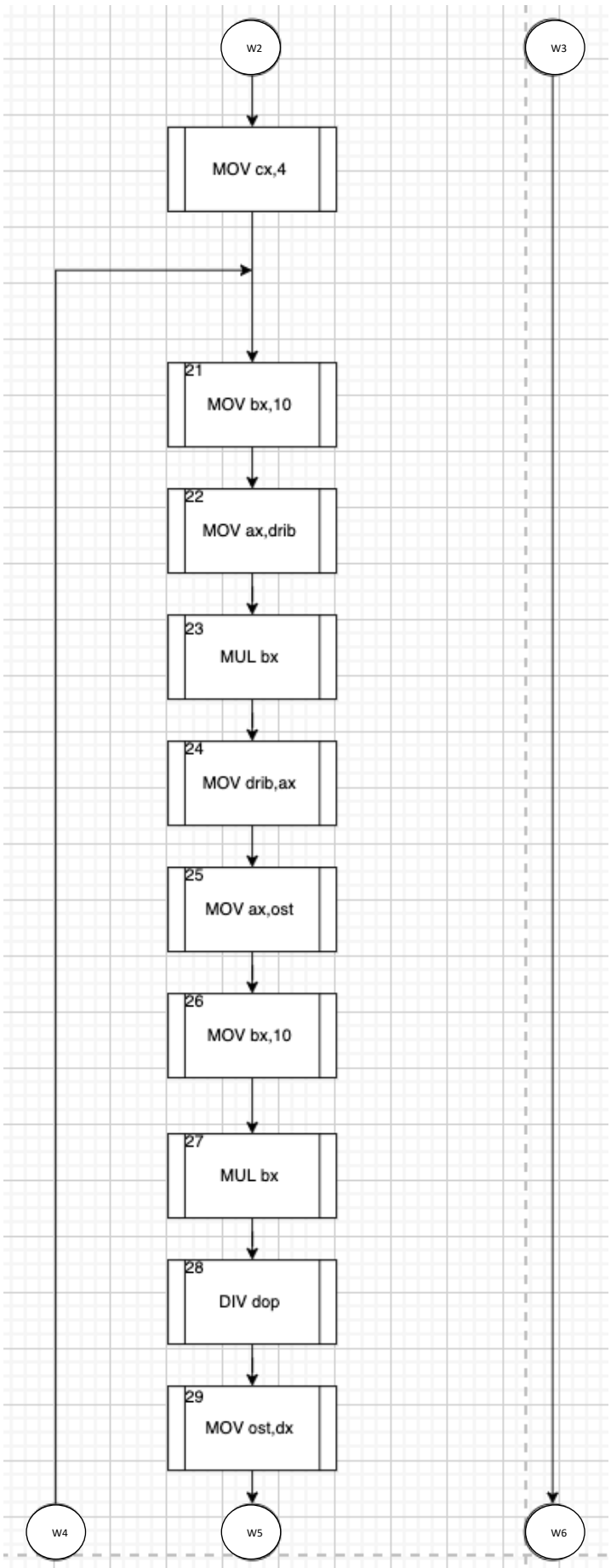


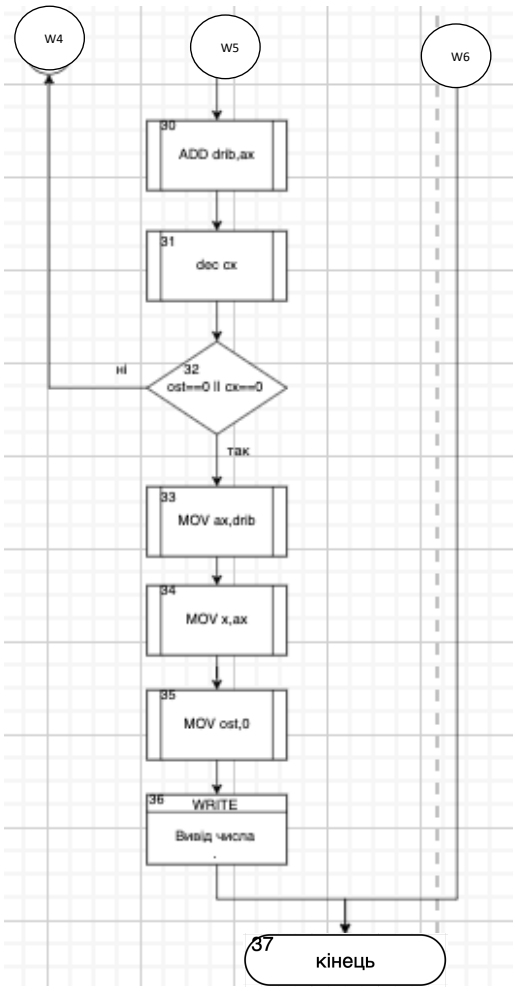
Рисунок 4.5 Схема функціонування процедури виведення числа



Продовження рисунку 4.5



Продовження рисунку 4.5



Продовження рисунку 4.5

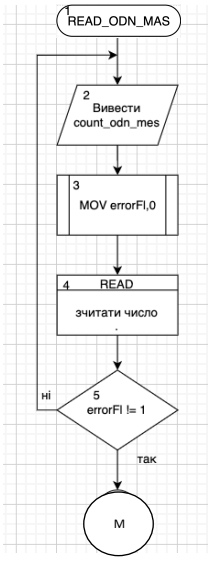
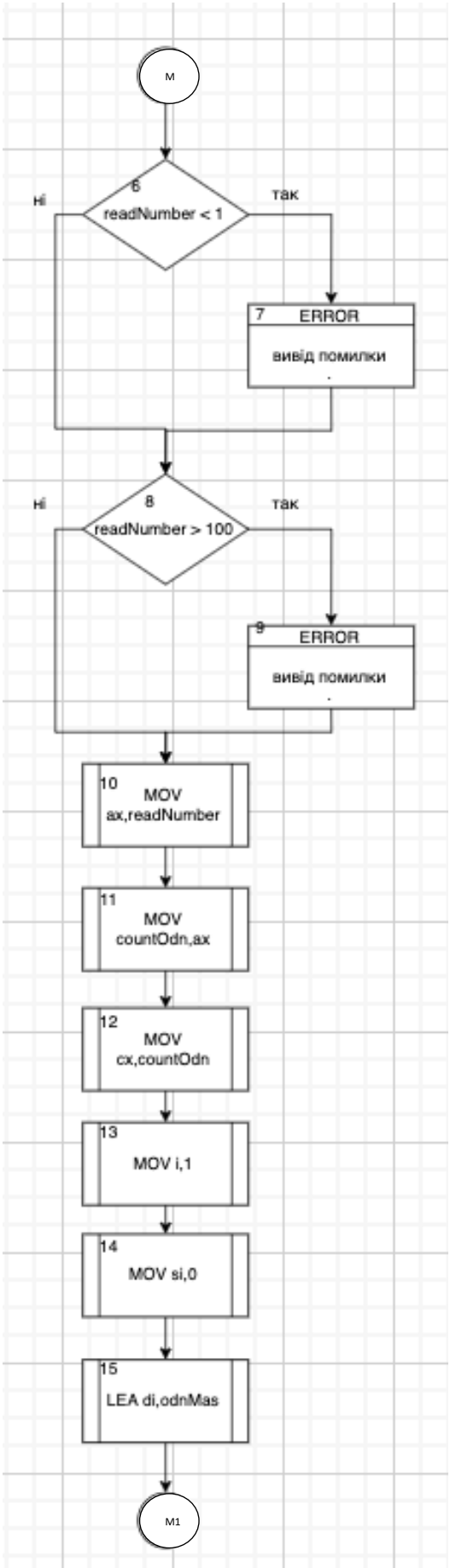
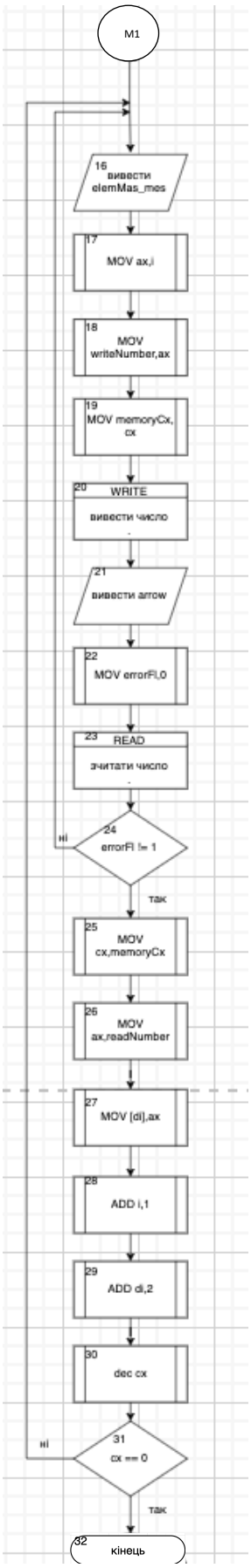


Рисунок 4.6 Схема функціонування процедури читання одновимірного масиву



Продовження рисунку 4.6



Продовження рисунку 4.6

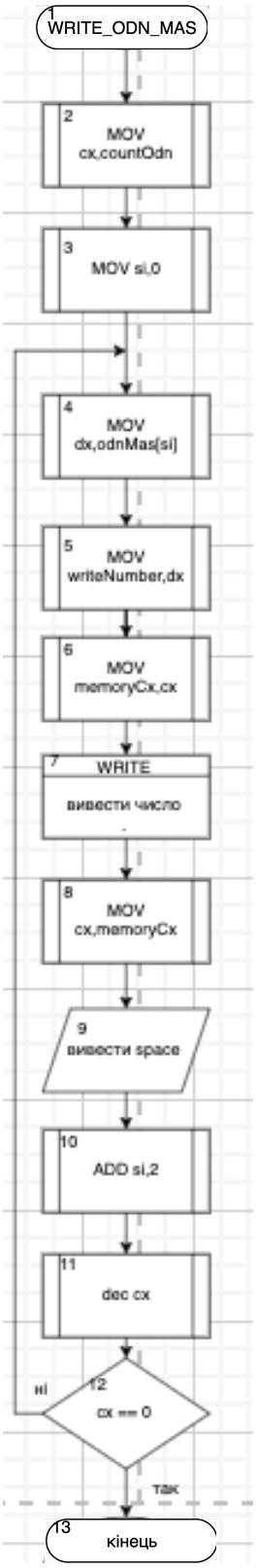


Рисунок 4.7 Схема функціонування процедури виведення одномірного масиву

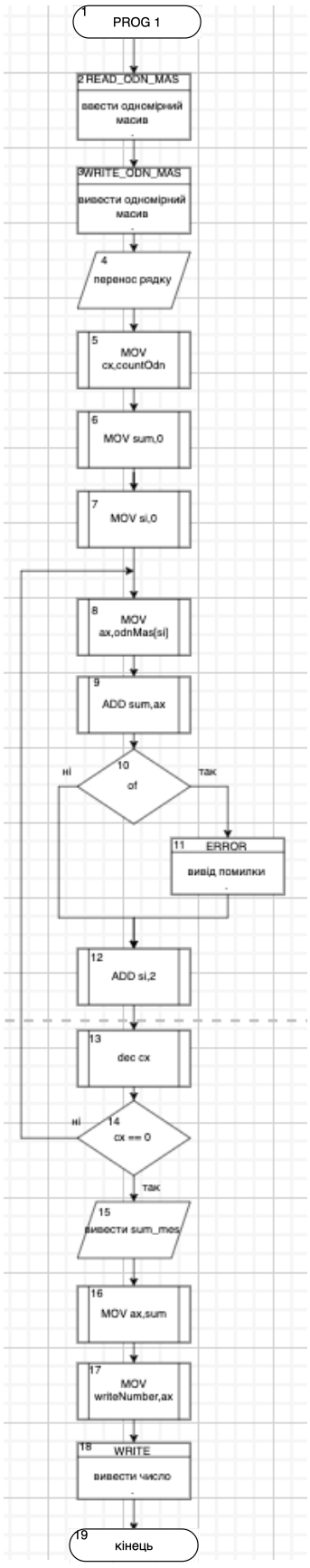


Рисунок 4.8 Схема функціонування процедури знаходження суми масиву

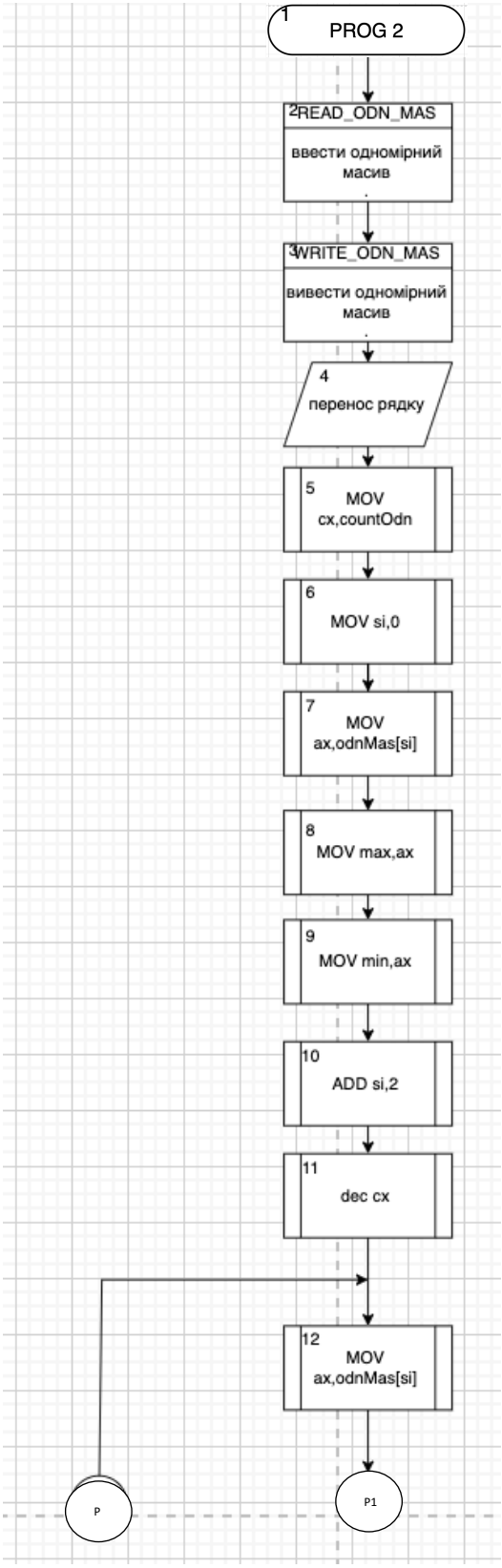
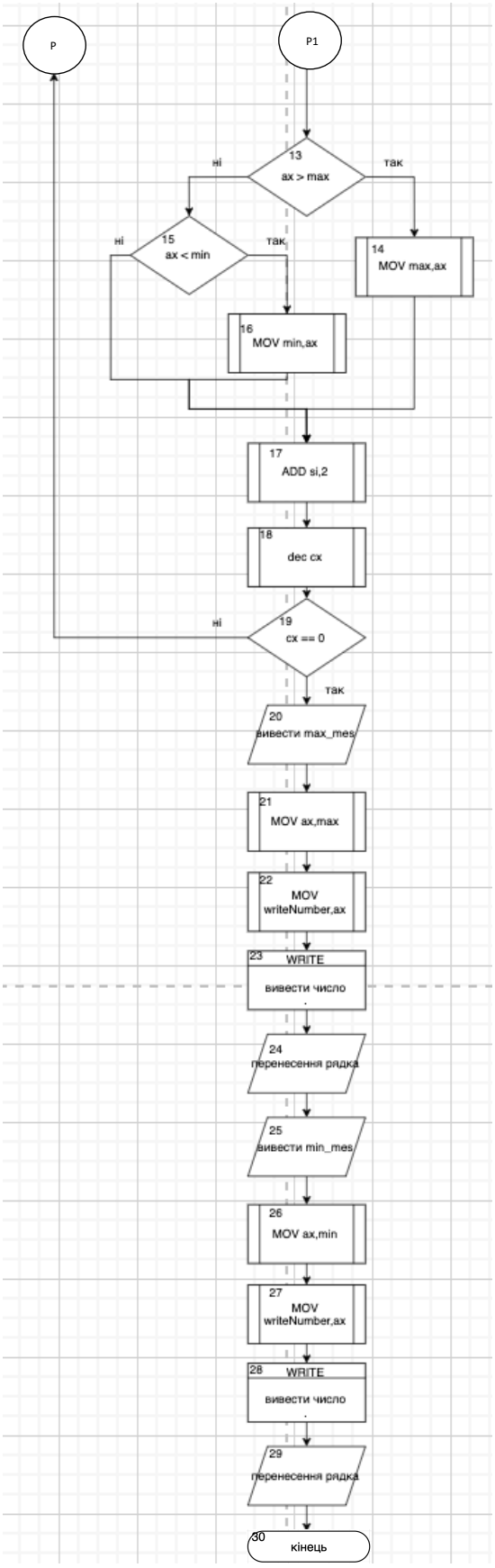


Рисунок 4.9 Схема функціонування процедури знаходження максимального і мінімального елементу



Продовження рисунку 4.9

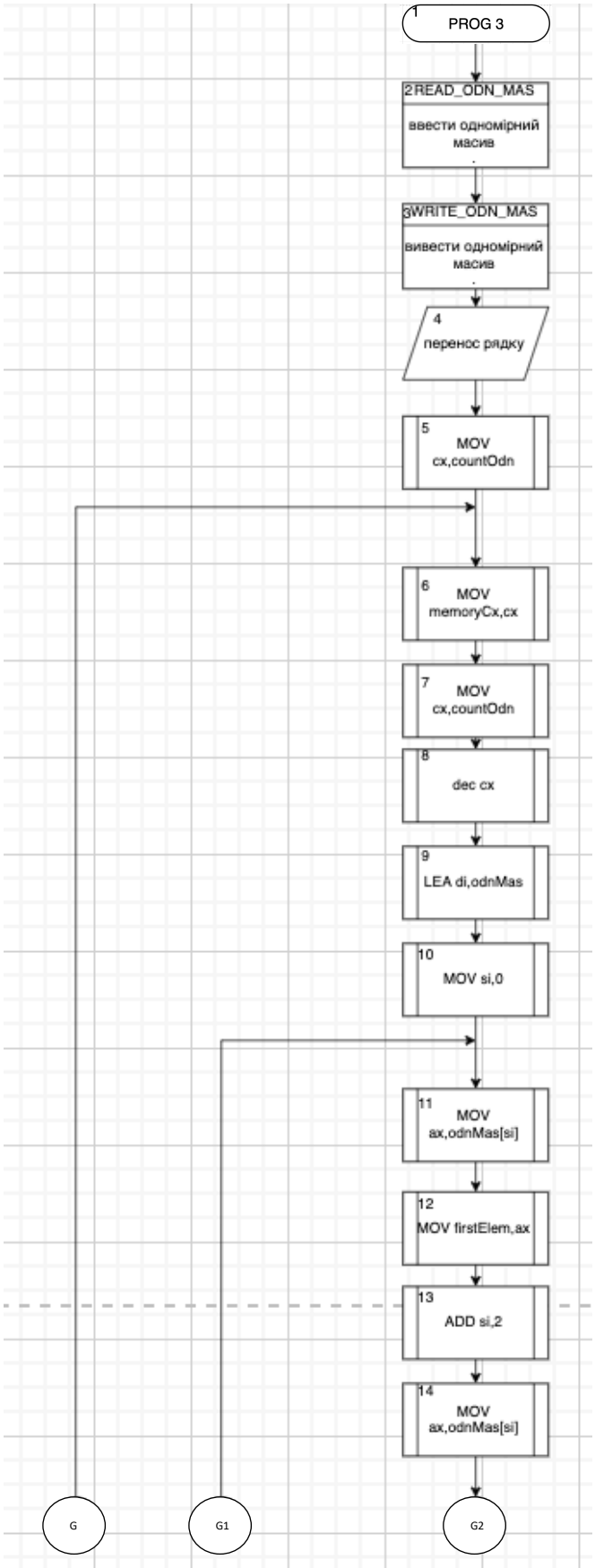
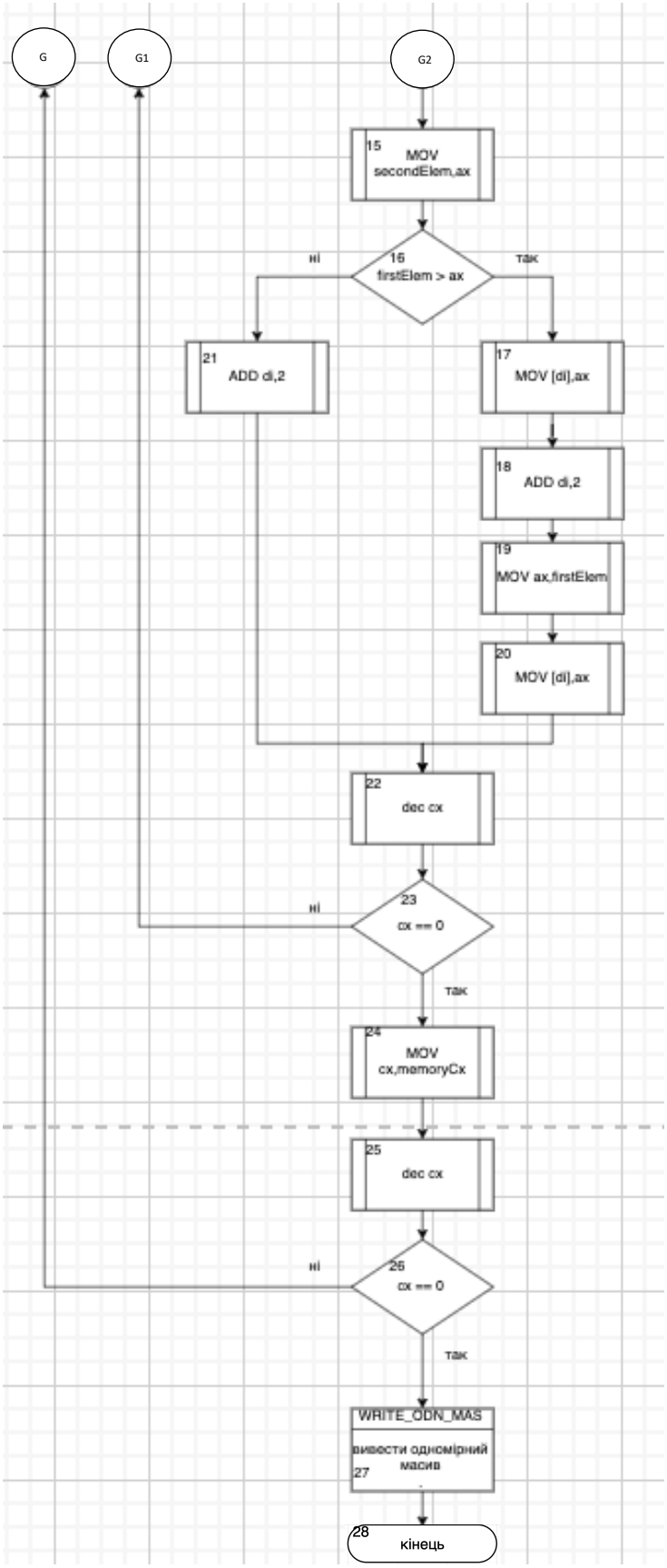


Рисунок 4.10 Схема функціонування процедури сортування масиву



Продовження рисунку 4.10

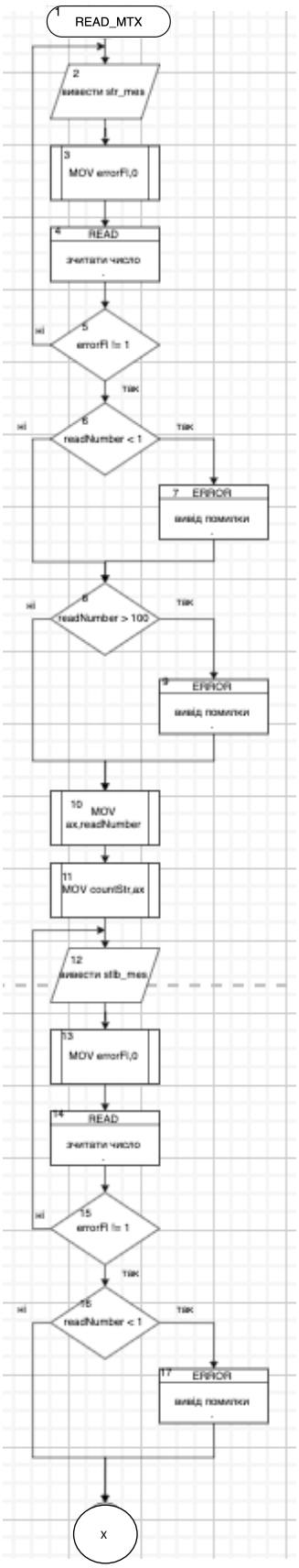
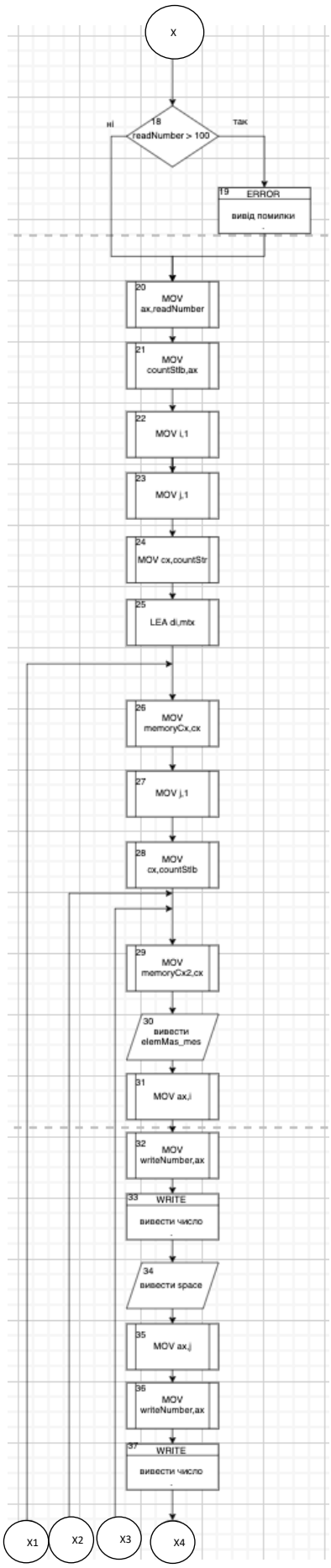
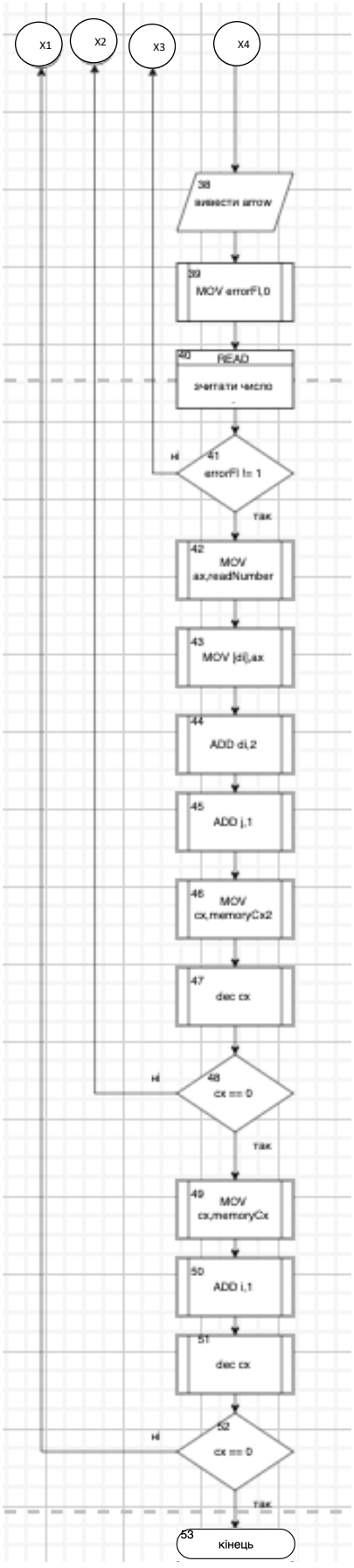


Рисунок 4.11 Схема функціонування процедури читання двовимірного масиву



Продовження рисунку 4.11



Продовження рисунку 4.11

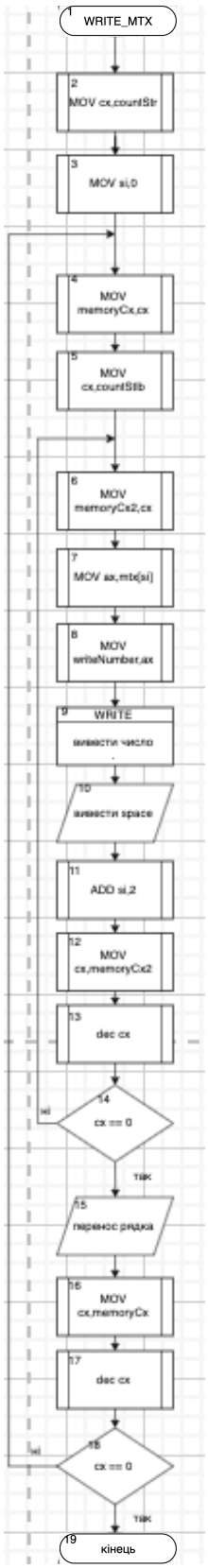
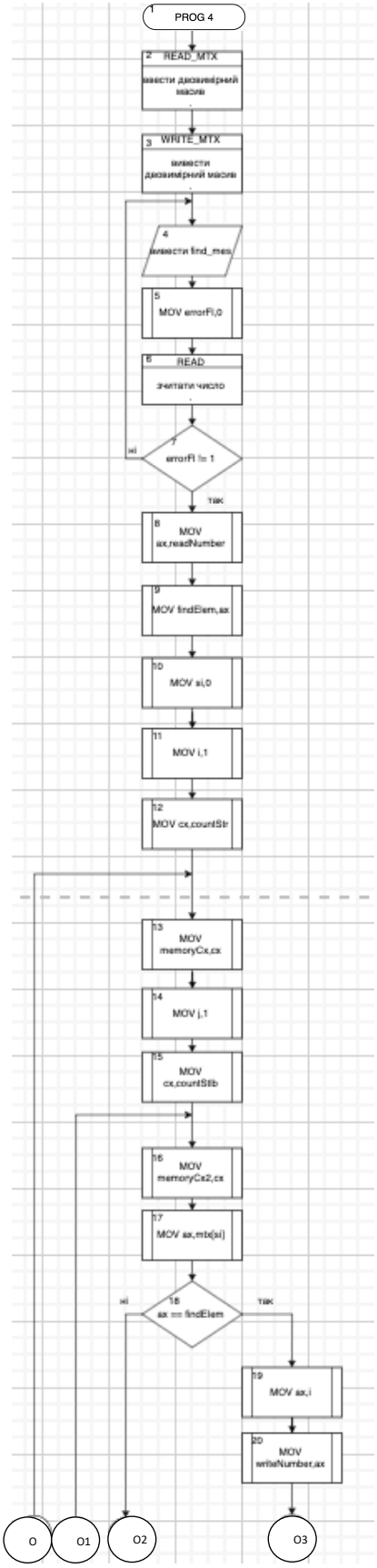
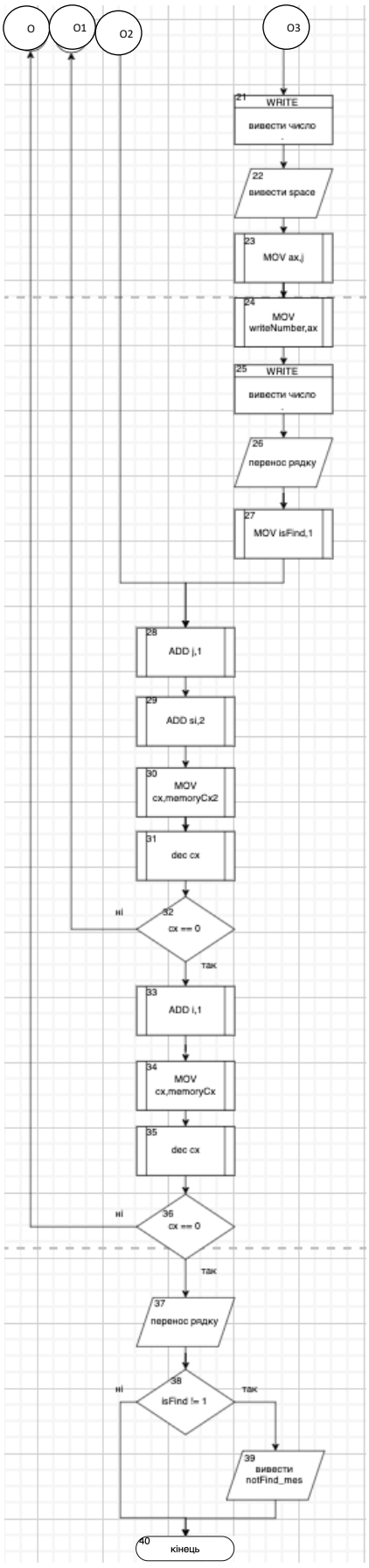


Рисунок 4.12 Схема функціонування процедури виведення двовимірного масиву



Рсунок 4.13 Схема функціонування процедури пошуку елементу у двовимірному масиві



Продовження рисунку 4.13

Приклади виконання програми

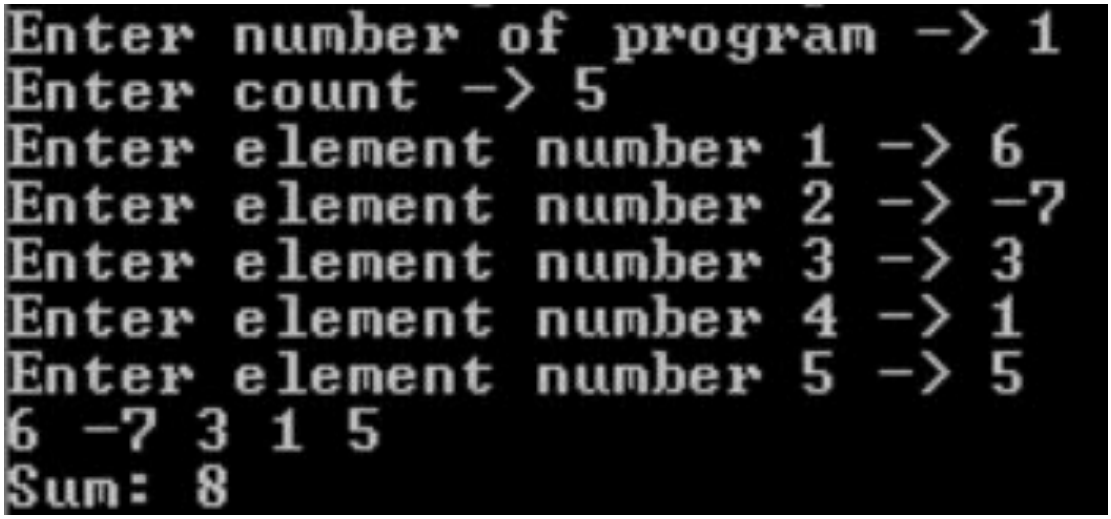


Рисунок 4.14 Приклад роботи програми

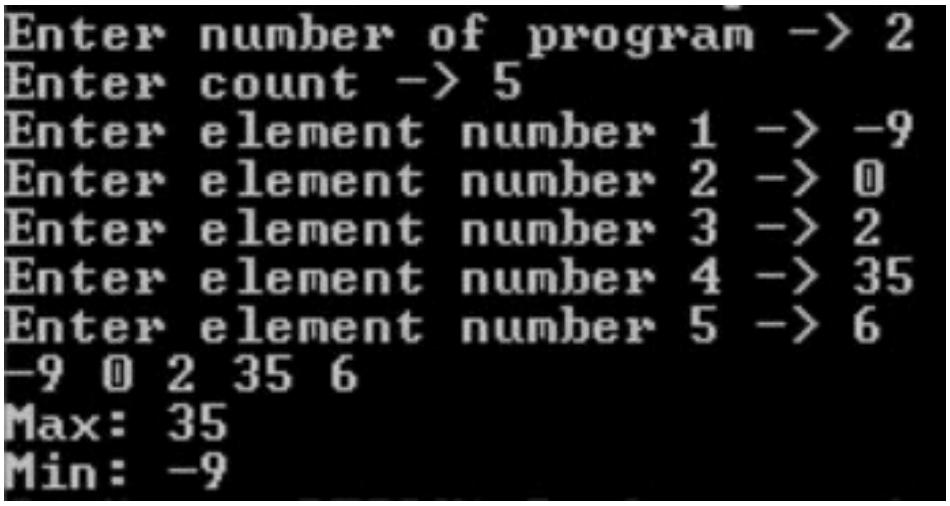


Рисунок 4.15 Приклад робот програми

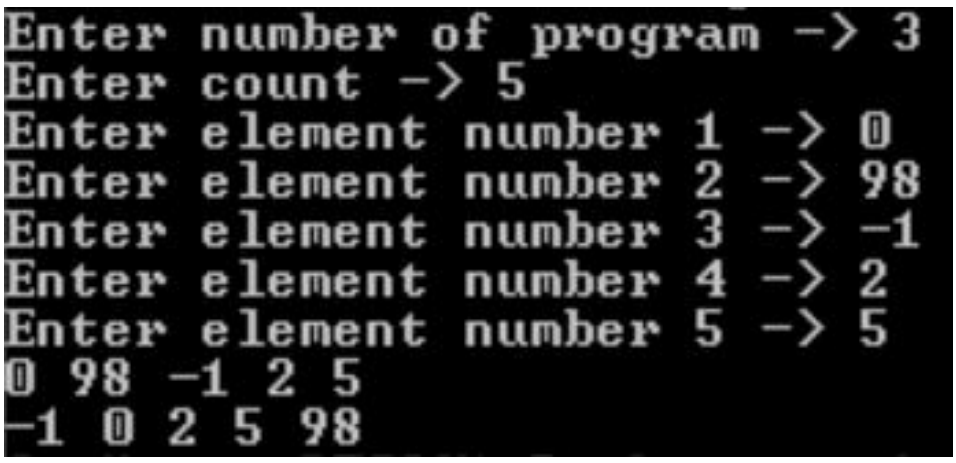


Рисунок 4.16 Приклад роботи програми

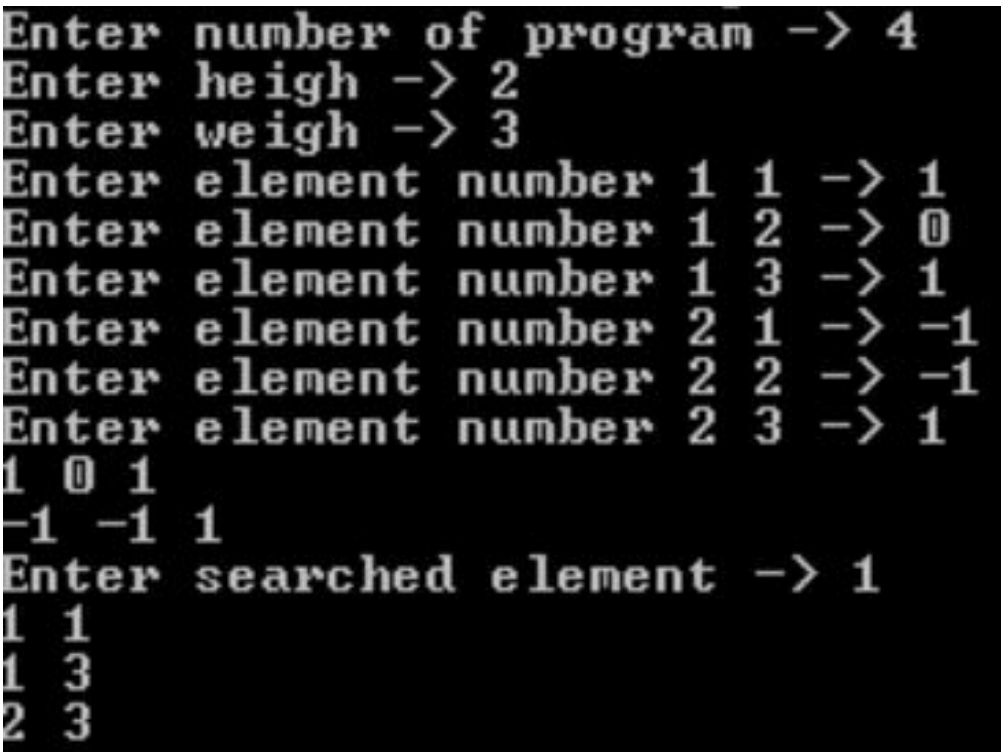


Рисунок 4.17 Приклад роботи програми

Висновок: Під час виконання комп’ютерного практикуму мною було створено програму, яка включає в себе 4 підпрограми:

1. 3 процедури одновимірного масиву: знаходження суми введеного масиву, знаходження максимального і мінімального елементу введеного масиву, сортування введеного масиву
2. знаходження елементу у введеному двовимірному масиві. Було протестовано програму.