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

Звіти до комп’ютерних практикумів дисципліни

«Системне програмне забезпечення»

Прийняв
доцент кафедри ІПІ
Лісовиченко О.І.
“26” травня 2023 р.

Виконав
Студент групи ІП-11
Прищепа В.С.

Київ-2023

Комп’ютерний практикум №5

Тема: макрозасоби мови асемблер.

Завдання:

Скласти програму на нижче наведені завдання:

1) переписати програму комп‘ютерного практикуму №2 з використанням одного макроса;

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.

Текст програми 1:

```
input_digit macro

;input a string
  lea dx, inarr
  mov ah, 10
  int 21h

;print a new line after the input
  mov al,10
  int 29h
  mov al,13
  int 29h
endm

STSEG SEGMENT PARA STACK "STACK"
DB 64 DUP("STACK")
STSEG ENDS

DSEG SEGMENT PARA PUBLIC "DATA"
inarr db 7,?,7 dup (" $")
input_tip db 13, 10, "Enter x in [-10922; 10922] =>> $"
is_negative db 0
num dw 0
digit dw 0
is_error db 0
error_message_1 db "Invalid symbol(s)!$"
error_message_2 db "Number out of diapason!$"
DSEG ENDS

CSEG SEGMENT PARA PUBLIC "CODE"
ASSUME CS:CSEG, DS:DSEG, SS:STSEG

main proc
```

```
;ds initialisation
  mov ax, dseg
  mov ds, ax

;showing a tip
  lea dx, input_tip
  mov ah, 9
  int 21h

;input of number
  input_digit
  call str_to_int
  cmp is_error, 1
  je raise_error_1
  cmp is_error, 2
  je raise_error_2

;3*num
  mov ax, num
  mov bx, 3
  imul bx
  jo raise_error_2
  mov num, ax

;printing of the result
  call result_print
  jmp end_program

;invalid symbol(s) error
  raise_error_1:
    LEA dx, error_message_1
    MOV ah,9
    INT 21h
    jmp end_program

;wrong number error
  raise_error_2:
    LEA dx, error_message_2
    MOV ah,9
    INT 21h

;program finishing
  end_program:
    mov AH, 4CH
    int 21H
    ret
```

```
main endp

str_to_int proc

;load the address of first element of the array
  mov si, offset inarr + 2
  mov cx, 0

;check if inarr is empty
    mov ax, 0
    mov al, inarr + 1
    cmp al, 0
    je error_1

;converting of the array into number
  convert_loop:

;check if it is the end of inarr
    mov ax, 0
    mov al, inarr + 1
    cmp al, cl
    je last

;set element of inarr to al
    mov al, [si]

;check if the character is between "0" and "9"
    cmp al, '0'
    jl check_minus
    cmp al, '9'
    jg error_1
    inc cx
    inc si

;jump for converting character to digit
    jmp convert_to_digit

;check character for "-"
  check_minus:
    cmp al, '-'
    jne error_1

;check if it is the first element in inarr
    cmp cx, 0
    jne error_1
    mov is_negative, 1
```

```
    inc cx
    inc si
    jmp convert_loop

;converting character to digit
  convert_to_digit:
    sub al, '0'
    mov digit, ax
    mov bx, 10
    mov ax, num
    mul bx
    jc error_2
    js error_2
    mov num, ax
    mov ax, digit
    add num, ax
    jc error_2
    js error_2
    jmp convert_loop

;Used wrong symbol(s)
  error_1:
    mov is_error, 1
    jmp finish

;Inputed number out of range
  error_2:
    mov is_error, 2
    jmp finish

;check if the number is negative
  last:
    cmp is_negative, 1
    jne finish
    neg num

  finish:
    ret
str_to_int endp

result_print proc

;check if result is negative
  mov bx, num
  or bx, bx
  jns m1
```

```
;if is negative, print '-' and make it positive
  mov al, '-'
  int 29h
  neg bx

;prepare to split number into digits
  m1:
    mov ax, bx
    xor cx, cx
    mov bx, 10

;split it into digits and write it down
  m2:
    xor dx, dx
    div bx
    add dl, '0'
    push dx
    inc cx
    test ax, ax
    jnz m2

;print these digits in correct order
  m3:
    pop ax
    int 29h
    loop m3
  ret
result_print endp

CSEG ENDS
END MAIN
```

Текст програми 2:

```
input_digit macro

;input a string
  lea dx, inar
  mov ah, 10
  int 21h

;print a new line after the input
  mov al,10
  int 29h
  mov al,13
  int 29h
```

```
endm

str_to_int macro inarr
local convert_loop, check_minus, convert_to_digit, error_1, error_2, last, finish

;load the address of first element of the array
  mov si, offset inarr + 2
  mov cx, 0

;check if inarr is empty
  mov ax, 0
  mov al, inarr + 1
  cmp al, 0
  je error_1

;converting of the array into number
  convert_loop:

;check if it is the end of inarr
    mov ax, 0
    mov al, inarr + 1
    cmp al, cl
    je last

;set element of inarr to al
    mov al, [si]

;check if the character is between "0" and "9"
    cmp al, '0'
    jl check_minus
    cmp al, '9'
    jg error_1
    inc cx
    inc si

;jump for converting character to digit
    jmp convert_to_digit

;check character for "-"
  check_minus:
    cmp al, '-'
    jne error_1

;check if it is the first element in inarr
    cmp cx, 0
    jne error_1
```

```
    mov is_negative, 1
    inc cx
    inc si
    jmp convert_loop

;converting character to digit
  convert_to_digit:
    sub al, '0'
    mov digit, ax
    mov bx, 10
    mov ax, num
    mul bx
    jc error_2
    js error_2
    mov num, ax
    mov ax, digit
    add num, ax
    jc error_2
    js error_2
    jmp convert_loop

;Used wrong symbol(s)
  error_1:
    mov is_error, 1
    jmp finish

;Inputed number out of range
  error_2:
    mov is_error, 2
    jmp finish

;check if the number is negative
  last:
    cmp is_negative, 1
    jne finish
    neg num

  finish:
endm

calc_res macro xpar
local x_zero, x_less, error, finishs

;compare xpar with 0
  cmp xpar, 0
  je x_zero
```

```
  jl x_less

;xpar>0 z=5*(4*xpar+7)/((xpar+1)*(xpar+2))
  mov ax, xpar
  add ax, 1
  jc error
  mov bx, ax
  add ax, 1
  jc error
  mul bx
  jc error
  mov denom, ax
  mov ax, xpar
  mov bx, 4
  mul bx
  jc error
  add ax, 7
  jc error
  mov bx, 5
  mul bx
  jc error
  mov bx, denom
  div bx
  mov num, ax
  mov ax, dx
  mov nom, ax
  jmp finishs

;xpar=0 z=5
  x_zero:
    mov ax, 5
    mov num, ax
    jmp finishs

;xpar<0 z=5*xpar^2/(1-xpar)
  x_less:
    mov ax, xpar
    neg ax
    jo error
    add ax, 1
    jc error
    mov denom, ax
    sub ax, 1
    mov bx, ax
    mul bx
```

```
    jc error
    mov bx, 5
    mul bx
    jc error
    mov bx, denom
    div bx
    mov num, ax
    mov ax, dx
    mov nom, ax
    jmp finishs

;xpar out of diapason
  error:
    mov is_error, 2

  finishs:
endm

result_print macro number
local m1, m2

;prepare to split number into digits
  mov ax, number
  xor cx, cx
  mov bx, 10

;split it into digits and write it down
  m1:
    xor dx, dx
    div bx
    add dl, '0'
    push dx
    inc cx
    test ax, ax
    jnz m1

;print these digits in correct order
  m2:
    pop ax
    int 29h
    loop m2
endm

exit macro

    mov AH, 4CH
```

```
    int 21H
endm

error_check macro
local raise_error_1, raise_error_2, continue

  cmp is_error, 1
  je raise_error_1
  cmp is_error, 2
  je raise_error_2
  jmp continue

;invalid symbol(s) error
  raise_error_1:
    LEA dx, error_message_1
    MOV ah,9
    INT 21h
    exit

;wrong number error
  raise_error_2:
    LEA dx, error_message_2
    MOV ah,9
    INT 21h
    exit

  continue:
endm

res_fraction macro
local frac, end_p

;printing of the result
  cmp num, 0
  je frac
  result_print num
  mov al, ' '
  int 29h

  cmp nom, 0
  jne frac
  jmp end_p

;printing of the fraction
  frac:
    result_print nom
```

```
    mov al, '/'
    int 29h
    result_print denom

  end_p:
endm

STSEG SEGMENT PARA STACK "STACK"
DB 64 DUP("STACK")
STSEG ENDS

DSEG SEGMENT PARA PUBLIC "DATA"
x dw 0
nom dw 0
denom dw 0
inar db 7,?,7 dup (" $")
input_tip db 13, 10, "Enter x in [-114;-1], {0} or [1;254] ==> $"
is_negative db 0
num dw 0
digit dw 0
is_error db 0
error_message_1 db "Invalid symbol(s)!$"
error_message_2 db "Number out of diapason!$"
DSEG ENDS

CSEG SEGMENT PARA PUBLIC "CODE"
ASSUME CS:CSEG, DS:DSEG, SS:STSEG

main proc

;ds initialisation
  mov ax, dseg
  mov ds, ax

;showing a tip
  lea dx, input_tip
  mov ah, 9
  int 21h

;input of number
  input_digit
  str_to_int inar
  error_check

;calculation
  mov ax, num
```

```
  mov x, ax
  calc_res x
  error_check
  res_fraction
  exit
  ret
main endp

CSEG ENDS
END MAIN
```

Особливості роботи:
В даній програмі ми передаємо параметри в деякі макроси. Покажемо, як це працює, на фрагментах lst-файла:

```
.....
    15                          str_to_int          macro inarr
    16                          local convert_loop, check_minus, convert_to_digit,
        error_1, error_2, last,       finish
    17
    18                          ;load the address of first       element       of the array
    19                            mov si, offset inarr + 2
    20                            mov cx, 0
    21
    22                          ;check if inarr is          empty
    23                            mov ax, 0
    24                            mov al, inarr + 1
    25                            cmp al, 0
    26                            je error_1
    27
    28                          ;converting of the       array into number
    29                            convert_loop:
    30
    31                          ;check if it is the end of       inarr
    32                                mov ax, 0
    33                                mov al, inarr + 1
    34                                cmp al, cl
    35                                je last
    36
    37                          ;set element of inarr to al
    38                                mov al, [si]
    39
    40                          ;check if the character is       between       "0" and       "9"
    41                                cmp al, '0'
    42                                jl check_minus
```

```
43                              cmp al, '9'
44                              jg error_1
45                              inc cx
46                              inc si
47
48                          ;jump for converting character to digit
49                              jmp convert_to_digit
50
51                          ;check character for "-"
52                            check_minus:
53                              cmp al, '-'
54                              jne error_1
55
56                          ;check if it is the first element in inarr
57                              cmp cx, 0
```



```
58                              jne error_1
59                              mov is_negative, 1
60                              inc cx
61                              inc si
62                              jmp convert_loop
63
64                          ;converting character to digit
65                            convert_to_digit:
66                              sub al, '0'
67                              mov digit, ax
68                              mov bx, 10
69                              mov ax, num
70                              mul bx
71                              jc error_2
72                              js error_2
73                              mov num, ax
74                              mov ax, digit
75                              add num, ax
76                              jc error_2
77                              js error_2
78                              jmp convert_loop
79
80                          ;Used wrong symbol(s)
```

```
    81                          error_1:
    82                              mov is_error, 1
    83                              jmp finish
    84
    85                          ;Inputed number out of range
    86                          error_2:
    87                              mov is_error, 2
    88                              jmp finish
    89
    90                          ;check if the number is negative
    91                          last:
    92                              cmp is_negative, 1
    93                              jne finish
    94                              neg num
    95
    96                          finish:
    97                          endm
.....
    301                                 str_to_int inar
1   302     001B  BE 0008r                  mov si, offset inar + 2
1   303     001E  B9 0000                   mov cx, 0
1   304     0021  B8 0000                   mov ax, 0
1   305     0024  A0 0007r                  mov al, inar + 1
1   306     0027  3C 00                 cmp al, 0
1   307     0029  74 4C                 je ??0003
1   308     002B                        ??0000:
1   309     002B  B8 0000                       mov ax, 0
1   310     002E  A0 0007r                      mov al, inar +      1
1   311     0031  3A C1                         cmp al, cl
1   312     0033  74 52                   je ??0005
1   313     0035  8A 04                   mov al, [si]
1   314     0037  3C 30                   cmp al, '0'
1   315     0039  7C 09                   jl ??0001
1   316     003B  3C 39                         cmp al, '9'
1   317     003D  7F 38                   jg ??0003
1   318     003F  41                      inc cx
1   319     0040  46                      inc si
1   320     0041  EB 13 90                      jmp ??0002
1   321     0044                        ??0001:
1   322     0044  3C 2D                         cmp al, '-'
1   323     0046  75 2F                   jne ??0003
1   324     0048  83 F9 00                      cmp cx, 0
1   325     004B  75 2A                         jne ??0003
1   326     004D  C6 06 0042r 01                mov is_negative, 1
1   327     0052  41                      inc cx
```

```
1   328         0053  46                    inc si
1   329         0054  EB D5                 jmp ??0000
1   330         0056                  ??0002:
1   331         0056  2C 30                 sub al, '0'
1   332         0058  A3 0045r              mov digit, ax
1   333         005B  BB 000A               mov bx, 10
1   334         005E  A1 0043r              mov ax, num
1   335         0061  F7 E3                 mul bx
1   336         0063  72 1A                 jc ??0004
1   337         0065  78 18                 js ??0004
1   338         0067  A3 0043r              mov num, ax
1   339         006A  A1 0045r              mov ax, digit
1   340         006D  01 06 0043r           add num, ax
1   341         0071  72 0C                 jc ??0004
1   342         0073  78 0A                 js ??0004
```



```
1   343         0075  EB B4                 jmp ??0000
1   344         0077                  ??0003:
1   345         0077  C6 06 0047r 01        mov is_error, 1
1   346         007C  EB 14 90              jmp ??0006
1   347         007F                  ??0004:
1   348         007F  C6 06 0047r 02        mov is_error, 2
1   349         0084  EB 0C 90              jmp ??0006
1   350         0087                  ??0005:
1   351         0087  80 3E 0042r 01        cmp is_negative, 1
1   352         008C  75 04                 jne ??0006
1   353         008E  F7 1E 0043r           neg num
1   354         0092                  ??0006:
.........
```

Тут, наприклад, ми можемо бачити, як транслюється макрос str_to_int в залежності від переданого параметра-рядка. У першому випадку було оголошено макрос, який приймав параметр inarr, у другому – програма викликала макрос із параметром inar, який був підставлений замість inarr, і через це в нас змінився фактичний код програми в lst файлі.

Текст програми 3:

```
input_digit macro
 mov ah, 9              ;showing a tip
```

```
 int 21h
 lea dx, inme          ;input a number
 mov ah, 10
 int 21h
 mov al,10             ;print a new line after the input
 int 29h
 mov al,13
 int 29h
endm

str_to_int macro inmes
local convert_loop, minus_check, convert_to_digit, add_digit, minimal_possible,
error_1, error_2, finn, done
 mov num, 0
 mov is_negative, 0
 mov si, offset inmes + 2 ;load the address of the first element
 mov ch, 0
convert_loop:
 mov ax, 0              ;check if it is the end of inmes
 mov al, inmes + 1
 cmp al, ch
 je finn
 mov al, [si]           ;set element of inmes to al
 cmp al, '0'
 jl minus_check         ;check if the character is between "0" and "9"
 cmp al, '9'
 jg error_1
 inc ch
 inc si
 jmp convert_to_digit   ;go to next element in inmes
minus_check:            ;check character for "-"
 cmp al, '-'
 jne error_1
 cmp ch, 0              ;check if it is the first element in inmes
 jne error_1
 mov is_negative, 1
 inc ch
 inc si
 jmp convert_loop
convert_to_digit:
 sub al, '0'
 mov digit, ax
 mov bx, 10
 mov ax, num
 mul bx
```

```
 jc error_2
 js error_2
 mov num, ax
 mov ax, digit
 cmp num, 32760
 je minimal_possible
add_digit:
 add num, ax
 jc error_2
 js error_2
 jmp convert_loop
minimal_possible:
 cmp is_negative, 0
 je add_digit
 neg num
 sub num, ax
 jo error_2
 jmp done
error_1:
 mov is_error, 1
 jmp done
error_2:
 mov is_error, 2
 jmp done
finn:
 cmp is_negative, 1       ;check if the number is negative
 jne done
 neg num
done:
 xor ch, ch
endm

result_print macro number
local m1, m2, m3
 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

result_print_dword macro summ
local m_1, m_2, m_3
.386
 mov ebx, summ
 or ebx, ebx
 jns m_1
 mov al, '-'
 int 29h
 neg ebx
m_1:
 mov eax, ebx
 xor cx, cx
 mov ebx, 10
m_2:
 xor edx, edx
 div ebx
 add dl, '0'
 push dx
 inc cx
 test eax, eax
 jnz m_2
m_3:
 pop ax
 int 29h
 loop m_3
endm

array_input macro
local elem_inp
.386
 lea dx, elements_tip          ;print a tip
 mov ah, 9
```

```
 int 21h
 mov al,10                ;print a new line
 int 29h
 mov al,13
 int 29h
 mov counter, 0
 mov di, 0
elem_inp:                 ;input loop
 lea dx, el_tip           ;formatted input array element
 mov ah, 9
 int 21h
 result_print counter
 lea dx, el_input_tip
 input_digit
 str_to_int inme
 error_check
 mov ax, num
 mov [arr+di], ax
 inc counter
 add di, 2
 mov cx, counter
 cmp cx, arr_size
 jne elem_inp
endm

array_sum macro summ
local sum_loop
.386
 lea dx, sum_print
 mov ah, 9
 int 21h
 xor dx, dx
 xor ax, ax
 mov cx, arr_size
 mov di, 0
sum_loop:
 mov ax, [arr+di]
 cwde
 add edx, eax
 add di, 2
 loop sum_loop
 mov summ, edx
endm

array_min_and_max macro
```

```
local read_num, set_new_min, set_new_max, exit
 mov ax, [arr]
 mov min, ax
 mov max, ax
 mov cx, arr_size
 sub cx, 1
 jcxz exit
 mov si, 2
read_num:
 mov ax, [arr+si]
 cmp ax, min
 jl set_new_min
 cmp ax, max
 jg set_new_max
 add si, 2
 loop read_num
 jmp exit
set_new_min:
 mov min, ax
 add si, 2
 dec cx
 jcxz exit
 jmp read_num
set_new_max:
 mov max, ax
 add si, 2
 dec cx
 jcxz exit
 jmp read_num
exit:
endm

array_sort macro
local outer_loop, inner_loop, continue, sort_done
 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, [arr+di]        ;compare this element and next one
 cmp ax, [arr+di+2]
 jng continue
```

```
 mov dx, [arr+di]        ;swap elements if next one less
 mov ax, [arr+di+2]
 mov [arr+di], ax
 mov [arr+di+2], dx
continue:
 add di, 2
 loop inner_loop
 inc si
 cmp si, arr_size
 jl outer_loop
sort_done:
endm

array_print macro
local show
 lea dx, sorted_arr_print
 mov ah, 09h
 int 21h
 mov di, 0
 mov si, 0
show:
 result_print [arr+si]
 mov al, ' '
 int 29h
 add si, 2
 inc di
 cmp di, arr_size
 jne show
endm

exit macro
  mov AH, 4CH
  int 21H
endm

error_check macro
local raise_error_1, raise_error_2, contin
  cmp is_error, 1
  je raise_error_1
  cmp is_error, 2
  je raise_error_2
  jmp contin
raise_error_1:                ;invalid symbol(s) error
  LEA dx, error_message_1
  MOV ah,9
```

```
  INT 21h
  exit
raise_error_2:                ;wrong number error
  LEA dx, error_message_2
  MOV ah,9
  INT 21h
  exit
contin:
endm

check_interval macro st, en, number
local final, con
 cmp number, st
 jle final
 cmp number, en
 jg final
 jmp con
final:
 LEA dx, error_message_2
 MOV ah,9
 INT 21h
 exit
con:
endm

STSEG SEGMENT PARA STACK "STACK"
DB 64 DUP("STACK")
STSEG ENDS

DSEG SEGMENT PARA PUBLIC "DATA"
arr_size dw 0
inme db 7,?,7 dup (" $")
el_tip db "The $"
el_input_tip db " element => $"
is_negative db 0
num dw 0
digit dw 0
is_error db 0
error_message_1 db "Invalid symbol(s)!$"
error_message_2 db "Number is out of diapason!$"
size_tip db "Enter the size of array (1-32) => $"
elements_tip db "Enter values in [-32768; 32767]:$"
arr dw 32 dup (?)
sum_print db "Total sum: $"
sum dd 0
```

```
min_print db 13,10, "Minimal element: $"
min dw 0
max_print db 13,10, "Maximal element: $"
max dw 0
sorted_arr_print 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
 lea dx, size_tip         ;input and convert size of array
 input_digit
 str_to_int inme
 error_check
 check_interval 0, 32, num
 mov ax, num
 mov arr_size, ax
 array_input              ;entering array elements
 error_check
 array_sum sum            ;calculate and print the sum
 result_print_dword sum
 array_min_and_max        ;find and print min and max element
 lea dx, min_print
 mov ah, 9
 int 21h
 result_print min
 lea dx, max_print
 mov ah, 9
 int 21h
 result_print max
 array_sort               ;sort array and print it
 array_print
 exit
 ret
main endp

CSEG ENDS
end main
```

Особливості роботи:

В даній програмі ми використовуємо мітки в деяких макроси. Ті макроси з мітками, які використовуються неодноразово, мають команду local. Покажемо, як вона працює, на фрагментах lst-файла:

```
.....
    364                                 str_to_int inme
1   365         001B  C7 06     0025r 0000    mov num, 0
1   366         0021  C6 06     0024r 00      mov is_negative, 0
1   367         0026  BE 0004r                mov si, offset inme +    2 ;load        the
address of the first element
1   368         0029  B5 00                   mov ch, 0
1   369         002B                        ??0000:
1   370         002B  B8 0000               mov ax, 0             ;check if itis the end of
inme
1   371         002E  A0 0003r              mov al, inme + 1
1   372         0031  3A C5                 cmp al, ch
1   373         0033  74 70                 je ??0007
1   374         0035  8A 04                 mov al, [si]                  ;set  element
        of inme       to al
1   375         0037  3C 30                 cmp al, '0'
1   376         0039  7C 0A                 jl ??0001        ;check ifthe character is
between "0" and "9"
1   377         003B  3C 39                 cmp al, '9'
1   378         003D  7F 56                 jg ??0005
1   379         003F  FE C5                 inc ch
1   380         0041  46                  inc si
1   381         0042  EB 14     90          jmp ??0002         ;go to         next
element in   inme
1   382         0045                      ??0001:           ;check character for "-"
1   383         0045  3C 2D                 cmp al, '-'
1   384         0047  75 4C                 jne ??0005
1   385         0049  80 FD     00          cmp ch, 0              ;check if itis the first
element in   inme
1   386         004C  75 47                 jne ??0005
1   387         004E  C6 06     0024r 01    mov is_negative, 1
1   388         0053  FE C5                 inc ch
1   389         0055  46                  inc si
1   390         0056  EB D3                 jmp ??0000
1   391         0058                      ??0002:
1   392         0058  2C 30                 sub al, '0'
1   393         005A  A3 0027r            mov digit, ax
1   394         005D  BB 000A             mov bx, 10
1   395         0060  A1 0025r            mov ax, num
1   396         0063  F7 E3                 mul bx
1   397         0065  72 36                 jc ??0006
```

```
1   398             0067  78 34                     js ??0006
1   399             0069  A3 0025r            mov num, ax
```



```
1   400             006C  A1 0027r            mov ax, digit
1   401             006F  81 3E  0025r 7FF8    cmp num, 32760
1   402             0075  74 0A                     je ??0004
1   403             0077                    ??0003:
1   404             0077  01 06  0025r              add num, ax
1   405             007B  72 20                     jc ??0006
1   406             007D  78 1E                     js ??0006
1   407             007F  EB AA                     jmp ??0000
1   408             0081                    ??0004:
1   409             0081  80 3E  0024r 00         cmp is_negative, 0
1   410             0086  74 EF                     je ??0003
1   411             0088  F7 1E  0025r              neg num
1   412             008C  29 06  0025r              sub num, ax
1   413             0090  70 0B                     jo ??0006
1   414             0092  EB 1C  90               jmp ??0008
1   415             0095                    ??0005:
1   416             0095  C6 06  0029r 01         mov is_error,       1
1   417             009A  EB 14  90               jmp ??0008
1   418             009D                    ??0006:
1   419             009D  C6 06  0029r 02         mov is_error,       2
1   420             00A2  EB 0C  90               jmp ??0008
1   421             00A5                    ??0007:
1   422             00A5  80 3E  0024r 01         cmp is_negative, 1          ;check if the
number       is negative
1   423             00AA  75 04                     jne ??0008
1   424             00AC  F7 1E  0025r              neg num
1   425             00B0                    ??0008:
1   426             00B0  32 ED                     xor ch, ch
.....
1   512                                      str_to_int inme
2   513             0159  C7 06  0025r 0000    mov num, 0
```

```
2  514          015F  C6 06     0024r 00       mov is_negative, 0
2  515          0164  BE 0004r          mov si, offset inme +    2 ;load        the
address of the first element
2  516          0167  B5 00                    mov ch, 0
2  517          0169                      ??0012:
2  518          0169  B8 0000             mov ax, 0             ;check if itis the end of
inme
2  519          016C  A0 0003r           mov al, inme + 1
2  520          016F  3A C5                    cmp al, ch
2  521          0171  0F 84     0084           je ??0019
2  522          0175  8A 04                    mov al, [si]                  ;set  element
       of inme       to al
2  523          0177  3C 30                    cmp al, '0'
2  524          0179  7C 0E     90 90          jl ??0013        ;check ifthe character is
between "0" and "9"
2  525          017D  3C 39                    cmp al, '9'
2  526          017F  7F 68     90 90          jg ??0017
2  527          0183  FE C5                    inc ch
2  528          0185  46                 inc si
2  529          0186  EB 18     90             jmp ??0014         ;go to         next
element in   inme
2  530          0189                     ??0013:          ;check character for "-"
2  531          0189  3C 2D                    cmp al, '-'
2  532          018B  75 5C     90 90          jne ??0017
2  533          018F  80 FD     00             cmp ch, 0             ;check if itis the first
element in   inme
2  534          0192  75 55     90 90          jne ??0017
2  535          0196  C6 06     0024r 01       mov is_negative, 1
2  536          019B  FE C5                    inc ch
2  537          019D  46                 inc si
2  538          019E  EB C9                    jmp ??0012
2  539          01A0                     ??0014:
2  540          01A0  2C 30                    sub al, '0'
2  541          01A2  A3 0027r           mov digit, ax
2  542          01A5  BB 000A            mov bx, 10
2  543          01A8  A1 0025r           mov ax, num
2  544          01AB  F7 E3                    mul bx
2  545          01AD  72 42     90 90          jc ??0018
2  546          01B1  78 3E     90 90          js ??0018
2  547          01B5  A3 0025r           mov num, ax
```

```
2   548             01B8  A1 0027r          mov ax, digit
2   549             01BB  81 3E     0025r 7FF8    cmp num, 32760
2   550             01C1  74 10     90 90         je ??0016
2   551             01C5                  ??0015:
2   552             01C5  01 06     0025r         add num, ax
2   553             01C9  72 26     90 90         jc ??0018
2   554             01CD  78 22     90 90         js ??0018
2   555             01D1  EB 96                   jmp ??0012
2   556             01D3                  ??0016:
2   557             01D3  80 3E     0024r 00      cmp is_negative, 0
2   558             01D8  74 EB                   je ??0015
2   559             01DA  F7 1E     0025r         neg num
2   560             01DE  29 06     0025r         sub num, ax
2   561             01E2  70 0D     90 90         jo ??0018
2   562             01E6  EB 1E     90            jmp ??001A
2   563             01E9                  ??0017:
2   564             01E9  C6 06     0029r 01      mov is_error,      1
2   565             01EE  EB 16     90            jmp ??001A
2   566             01F1                  ??0018:
2   567             01F1  C6 06     0029r 02      mov is_error,      2
2   568             01F6  EB 0E     90            jmp ??001A
2   569             01F9                  ??0019:
2   570             01F9  80 3E     0024r 01      cmp is_negative, 1          ;check if the
number       is negative


Turbo Assembler    Version 4.0     05/08/23 11:14:08          Page 11
Lab5_3.asm


2   571             01FE  75 06     90 90         jne ??001A
2   572             0202  F7 1E     0025r         neg num
2   573             0206                  ??001A:
2   574             0206  32 ED                   xor ch, ch
.........
```

Тут, наприклад, ми можемо бачити, як транслюється макрос str_to_int, що має в собі команду local. Фактичні назви його міток на кожному виклику набувають унікальних значень-чисел.

Схема роботи програми 1:

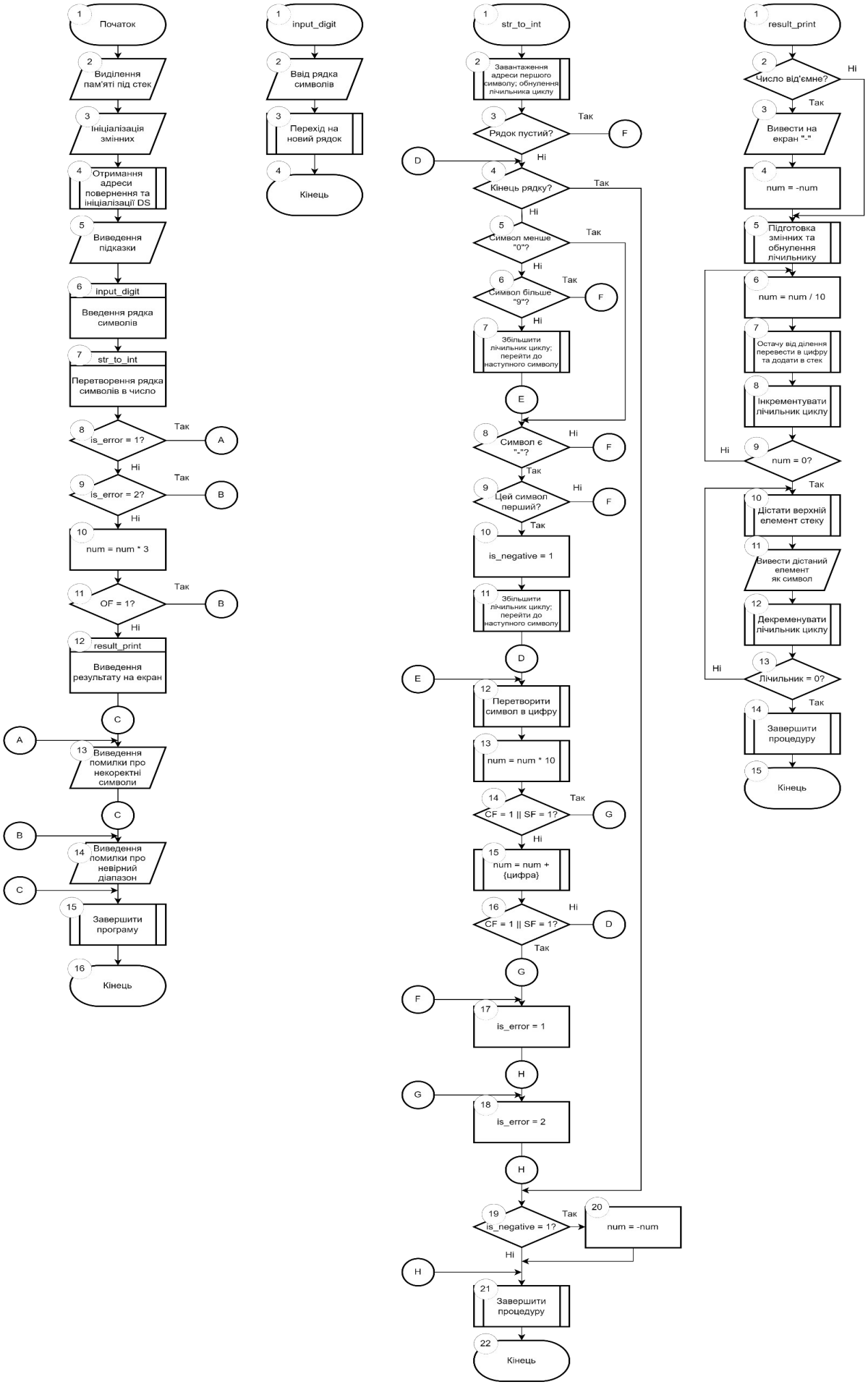

1 Початок
2 Виділення пам'яті під стек
3 Ініціалізація змінних
4 Отримання адреси повернення та ініціалізації DS
5 Виведення підказки
6 input_digit
Введення рядка символів
7 str_to_int
Перетворення рядка символів в число
8 is_error = 1?
Так
A
Ні
9 is_error = 2?
Так
B
Ні
10 num = num * 3
11 OF = 1?
Так
B
Ні
12 result_print
Виведення результату на екран
C
A
13 Виведення помилки про некоректні символи
C
B
14 Виведення помилки про невірний діапазон
C
15 Завершити програму
16 Кінець
1 input_digit
2 Ввід рядка символів
3 Перехід на новий рядок
4 Кінець
1 str_to_int
2 Завантаження адреси першого символу; обнулення лічильника циклу
3 Рядок пустий?
Так
F
Ні
D
4 Кінець рядку?
Так
Ні
5 Символ менше "0"?
Так
Ні
6 Символ більше "9"?
Так
F
Ні
7 Збільшити лічильник циклу; перейти до наступного символу
E
8 Символ є "-"?
Ні
F
Так
9 Цей символ перший?
Ні
F
Так
10 is_negative = 1
11 Збільшити лічильник циклу; перейти до наступного символу
D
E
12 Перетворити символ в цифру
13 num = num * 10
14 CF = 1 || SF = 1?
Так
G
Ні
15 num = num + {цифра}
16 CF = 1 || SF = 1?
Ні
D
Так
G
F
17 is_error = 1
H
G
18 is_error = 2
H
19 is_negative = 1?
Так
20 num = -num
Ні
H
21 Завершити процедуру
22 Кінець
1 result_print
2 Число від'ємне?
Ні
Так
3 Вивести на екран "-"
4 num = -num
5 Підготовка змінних та обнулення лічильнику
6 num = num / 10
7 Остачу від ділення перевести в цифру та додати в стек
8 Інкрементувати лічильник циклу
9 num = 0?
Ні
Так
10 Дістати верхній елемент стеку
11 Вивести дістаний елемент як символ
12 Декрементувати лічильник циклу
13 Лічильник = 0?
Ні
Так
14 Завершити процедуру
15 Кінець

Схема роботи програми 2:

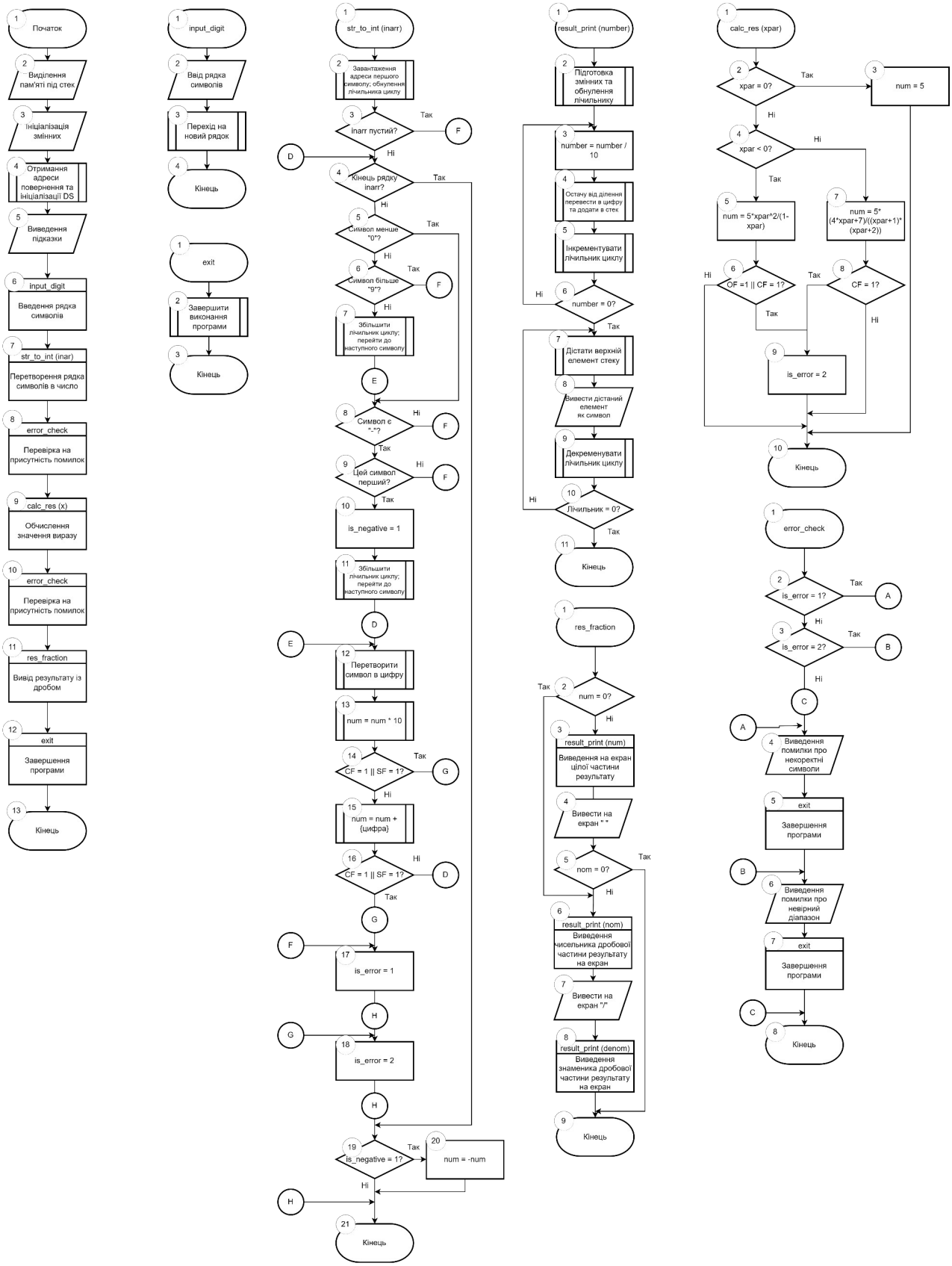
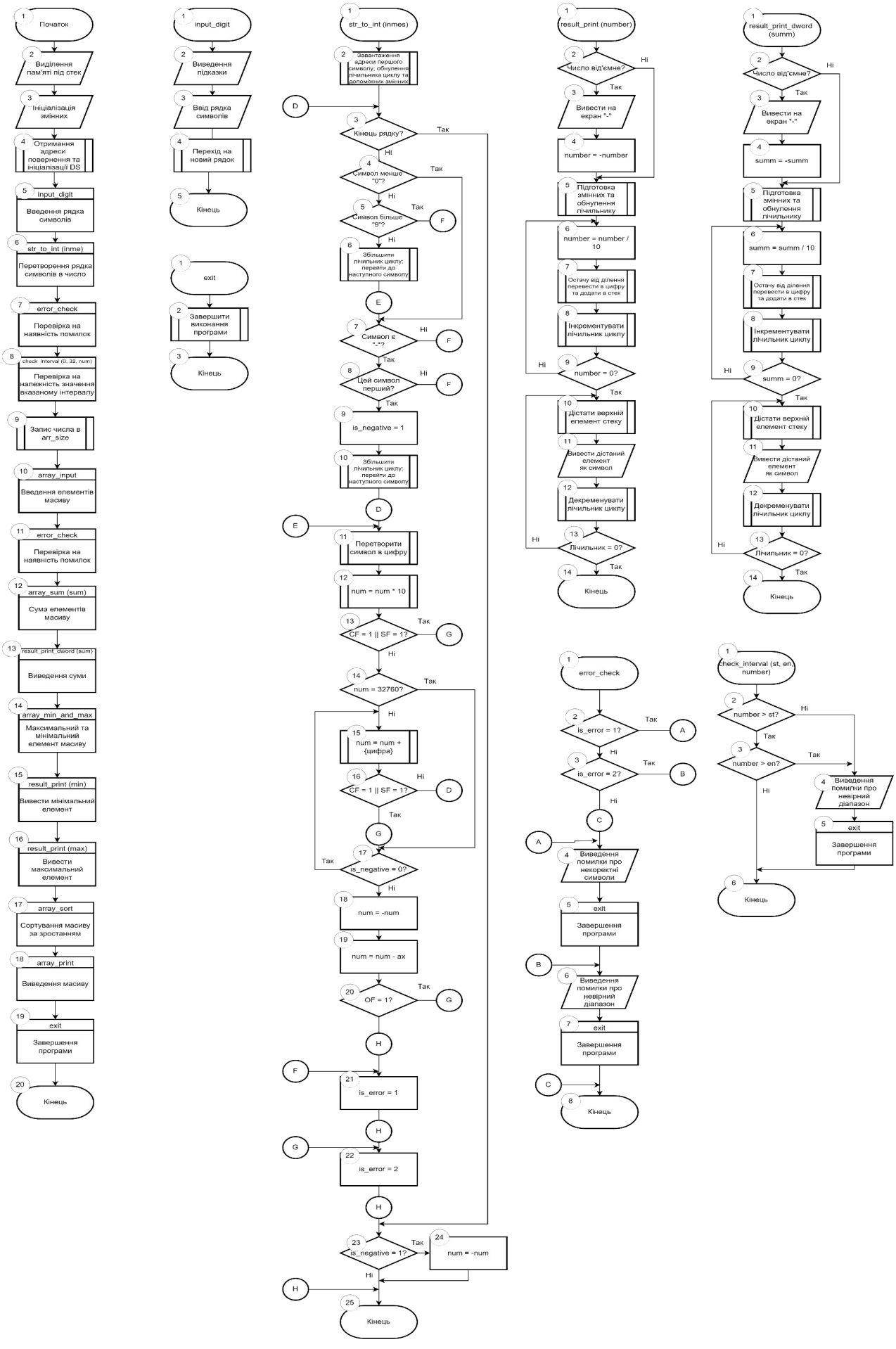
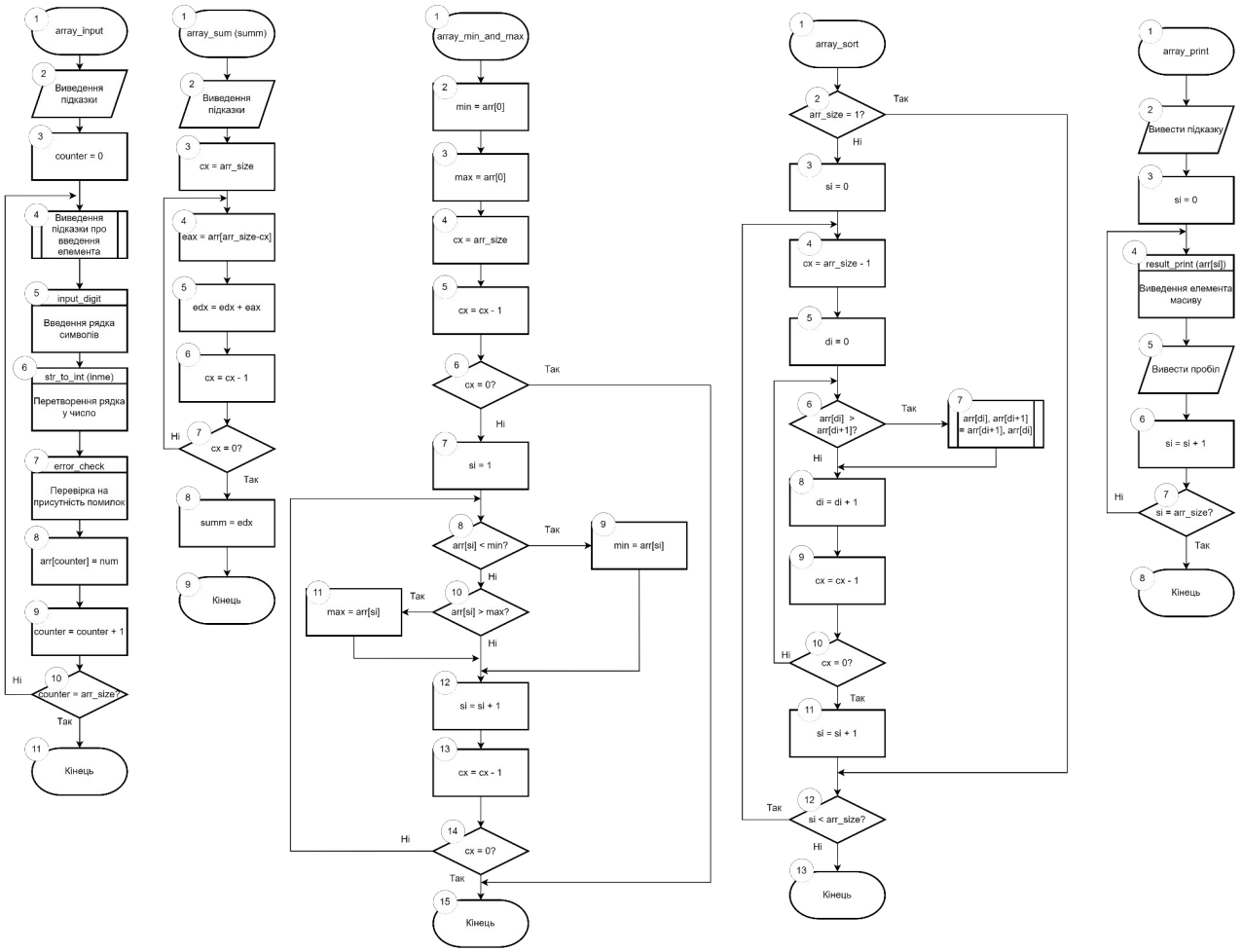


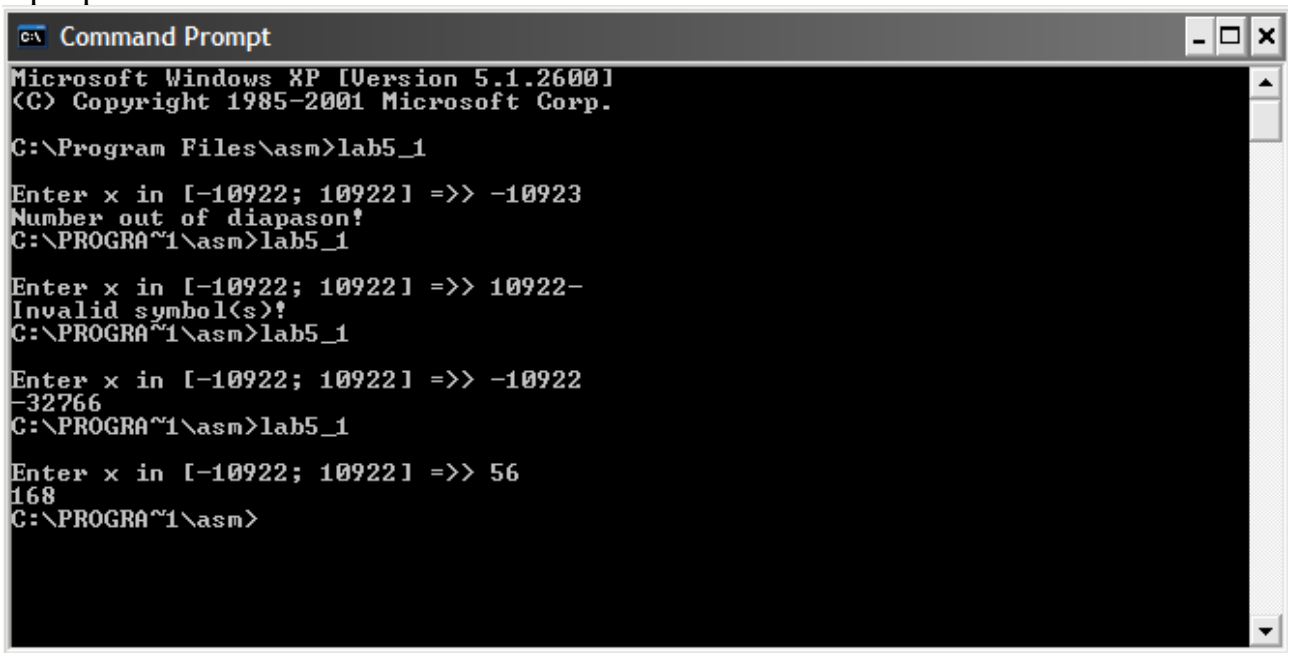
Схема роботи програми 3:


Початок
Виділення пам'яті під стек
Ініціалізація змінних
Отримання адреси повернення та ініціалізації DS
input_digit
Введення рядка символів
str_to_int (inme)
Перетворення рядка символів в число
error_check
Перевірка на наявність помилок
check_interval (0, 32, num)
Перевірка на належність значення вказаному інтервалу
Запис числа в arr_size
array_input
Введення елементів масиву
error_check
Перевірка на наявність помилок
array_sum (sum)
Сума елементів масиву
result_print_dword (sum)
Виведення суми
array_min_and_max
Максимальний та мінімальний елемент масиву
result_print (min)
Вивести мінімальний елемент
result_print (max)
Вивести максимальний елемент
array_sort
Сортування масиву за зростанням
array_print
Виведення масиву
exit
Завершення програми
Кінець
input_digit
Виведення підказки
Ввід рядка символів
Перехід на новий рядок
Кінець
exit
Завершити виконання програми
Кінець
str_to_int (inmes)
Завантаження адреси першого символу; обнулення лічильника циклу та допоміжних змінних
Кінець рядку?
Так
Ні
Символ менше "0"?
Символ більше "9"?
Збільшити лічильник циклу; перейти до наступного символу
Символ є "-"?
Цей символ перший?
is_negative = 1
Перетворити символ в цифру
num = num * 10
CF = 1 || SF = 1?
num = 32760?
num = num + {цифра}
is_negative = 0?
num = -num
num = num - ax
OF = 1?
is_error = 1
is_error = 2
is_negative = 1?
num = -num
result_print (number)
Число від'ємне?
Вивести на екран "-"
number = -number
Підготовка змінних та обнулення лічильнику
number = number / 10
Остачу від ділення перевести в цифру та додати в стек
Інкрементувати лічильник циклу
number = 0?
Дістати верхній елемент стеку
Вивести дістаний елемент як символ
Декрементувати лічильник циклу
Лічильник = 0?
result_print_dword (summ)
Число від'ємне?
Вивести на екран "-"
summ = -summ
Підготовка змінних та обнулення лічильнику
summ = summ / 10
Остачу від ділення перевести в цифру та додати в стек
Інкрементувати лічильник циклу
summ = 0?
Дістати верхній елемент стеку
Вивести дістаний елемент як символ
Декрементувати лічильник циклу
Лічильник = 0?
error_check
is_error = 1?
is_error = 2?
Виведення помилки про некоректні символи
exit
Завершення програми
Виведення помилки про невірний діапазон
exit
Завершення програми
check_interval (st, en, number)
number > st?
number > en?
Виведення помилки про невірний діапазон
exit
Завершення програми
Кінець



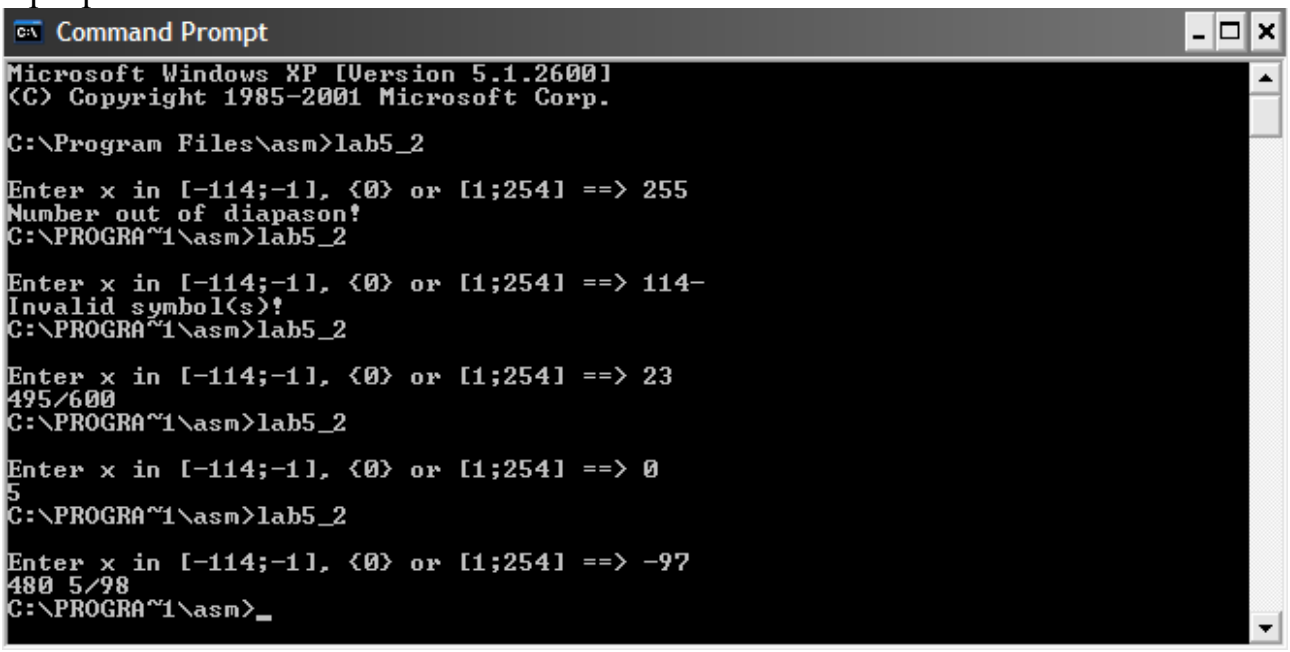
Приклади роботи програм:

Програма 1:



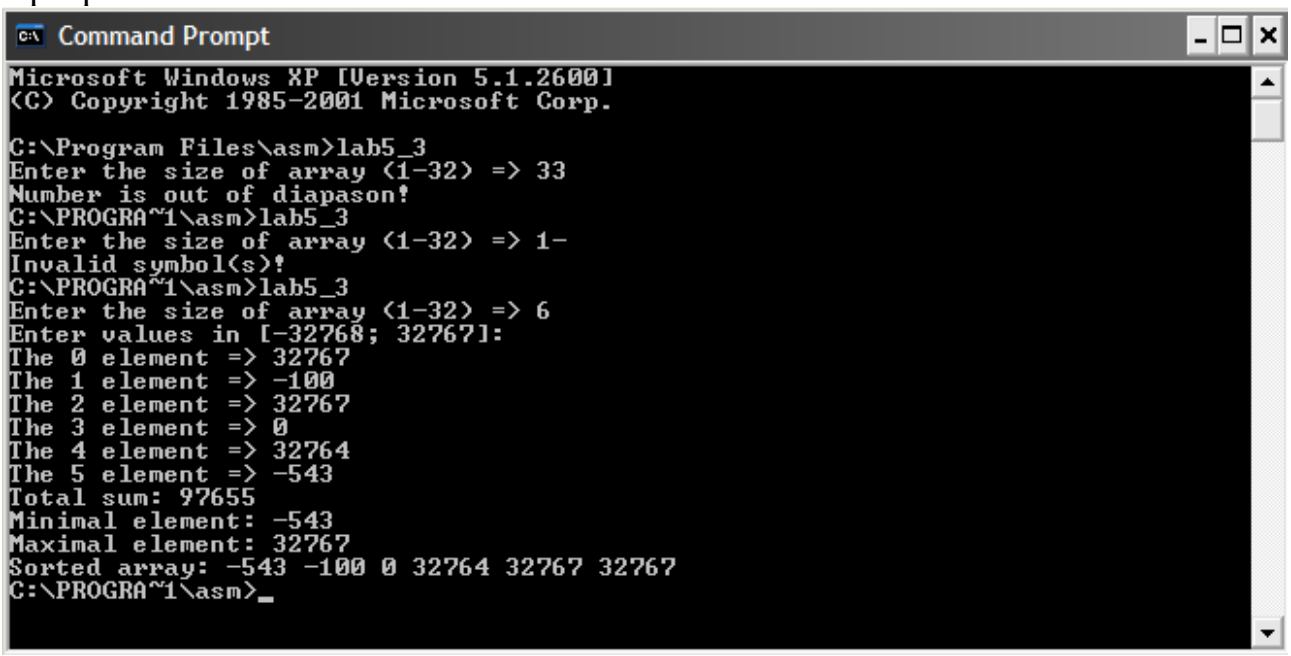
У першому виклику введене значення було поза дозволеного діапазону, в другому — було введене не число, а в третьому та четвертому — були введені правильні числа і програма вивела не повідомлення про помилку, а результат.

Програма 2:



У першому та другому викликах були введені значення поза діапазоном та не число відповідно, що призвело до помилки. В останніх трьох викликах були введені значення для всіх випадків, які передбаченні програмою.

Програма 3:



Перші два рази програма вивела помилки «Число поза дозволеного діапазону» чи «Використано невірні символи». Третього разу було продемонстровано, як програма працює при умові введення правильних значень. Спочатку було введено користувачем розмір та значення масиву, а далі програма підрахувала суму, знайшла мінімальний та максимальний елементи масиву і відсортувала масив за зростанням.

Висновок:

Складено програми на нижче наведені завдання (код і приклади виконання наведені нижче):
1) переписано програму комп‘ютерного практикуму №2 з використанням одного макросу (input_digit);
2) переписано програму комп‘ютерного практикуму №3 з використанням макросів та передачею параметрів в них (всі процедури перероблені в макроси, більшість з них приймає параметри). Особливості передачі параметрів у макроси продемонстровано в звіті;
3) переписана програма про масив комп‘ютерного практикуму №4 з використанням макросів та залученням міток в тілі макросу (всі процедури переписані в макроси, більшість з них має мітки). Особливості міток у макросах продемонстровано в звіті.