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 Виконав**

**доцент кафедри ІПІ Студент групи ІП-11**

**Лісовиченко О.І. Головня О.Р.**

**“30” травня 2023 р.**

Київ – 2023

**Комп’ютерний практикум №4**

**Тема**: Масиви

**Завдання:**

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.

Програма повинна мати захист від некоректного введення вхідних даних (символи, переповнення і т.і.)

Код програми:

sseg segment para stack 'stack'

db 64 dup ( 'STACK' )

sseg ends

dseg segment para public 'data'

enter\_arr\_size\_msg db 0dh, 0ah, 'enter array length (2-20): $'

input\_error\_message db 0dh, 0ah, 'the number you entered is too large$',10

min\_error\_message db 0dh, 0ah, 'array has to have at least 2 elements $'

max\_error\_message db 0dh, 0ah, 'array can`t have more than 20 elements $'

wrong\_character\_message db 0dh, 0ah, 'wrong character$'

empty\_input\_message db 0dh, 0ah, 'empty input$'

enter\_element\_message db 0dh, 0ah, 'enter element in (-32768; 32767]: $'

space db ' $'

array\_elements\_message db 0dh, 0ah, 'Your array : $'

max\_element\_message db 0dh, 0ah, 'Max: $'

sum\_message db 0dh, 0ah, 'Sum of elements: $'

sorted\_message db 0dh, 0ah, 'Sorted successfully $'

overflow\_error db 0dh, 0ah, 'There was an overflow when adding(-32768;32767]$'

array dw 20 dup(0)

sizeArr dw 0

num dw 0

temp dw 0

is\_negative dw 0

minArrayLength dw 2

maxArrayLength dw 20

min\_element dw 0

max\_element dw 0

dseg ends

cseg segment para public 'code'

print macro string

push ax

lea dx,string

mov ah,09h

int 21h

pop ax

endm

main proc

assume cs: cseg, ds: dseg, ss: sseg

mov ax, dseg

mov ds, ax

call enter\_array

call print\_array

call sum\_of\_array

call find\_max\_element

call sort

call print\_array

main endp

exit proc

mov ax, 4c00h

int 21h

ret

exit endp

enter\_array proc

print enter\_arr\_size\_msg

call read\_digit

mov ax, num

mov sizeArr, ax

call check\_array\_length

mov cx, ax

xor bx, bx

lea si, array

read\_array\_loop:

push cx

print enter\_element\_message

call read\_digit1

mov ax, num

mov [si], ax

add si, 2

pop cx

loop read\_array\_loop

ret

enter\_array endp

read\_digit proc

mov num,0

mov bx, 10

mov cx, 5

mov is\_negative,0

read:

xor ax, ax

mov ah, 01h

int 21h

cmp al, 13

je stop

cmp al, 48

jl check\_sign

cmp al, 57

ja wrong\_character

sub al, '0'

sub ah, ah

mov temp, ax

mov ax, num

imul bx

jo input\_error

add ax, temp

jo input\_error

mov num, ax

loop read

cmp is\_negative, 1

je make\_negative

ret

stop:

cmp cx, 5

je empty\_input

mov cx, 0

cmp is\_negative, 1

je make\_negative

ret

make\_negative:

neg num

ret

check\_sign:

cmp al, '-'

jne wrong\_character

cmp cx, 5

jne wrong\_character

mov is\_negative, 1

jmp read

input\_error:

print input\_error\_message

call exit

wrong\_character:

print wrong\_character\_message

call exit

empty\_input:

print empty\_input\_message

call exit

read\_digit endp

read\_digit1 proc

mov num,0

mov bx, 10

mov cx, 5

mov is\_negative,0

read1:

xor ax, ax

mov ah, 01h

int 21h

cmp al, 13

je stop1

cmp al, 48

jl check\_sign1

cmp al, 57

ja wrong\_character1

sub al, '0'

sub ah, ah

mov temp, ax

mov ax, num

imul bx

jo input\_error1

add ax, temp

jo input\_error1

mov num, ax

loop read1

cmp is\_negative, 1

je make\_negative1

ret

stop1:

cmp cx, 5

je empty\_input1

mov cx, 0

cmp is\_negative, 1

je make\_negative1

ret

make\_negative1:

neg num

ret

check\_sign1:

cmp al, '-'

jne wrong\_character1

cmp cx, 5

jne wrong\_character1

mov is\_negative, 1

jmp read1

input\_error1:

print input\_error\_message

print enter\_element\_message

jmp read\_digit1

wrong\_character1:

print wrong\_character\_message

print enter\_element\_message

jmp read\_digit1

empty\_input1:

print empty\_input\_message

print enter\_element\_message

jmp read\_digit1

read\_digit1 endp

check\_array\_length proc

xor ax,ax

mov ax,sizeArr

cmp ax, minArrayLength

jl min\_error

cmp ax, maxArrayLength

jg max\_error

ret

min\_error:

print min\_error\_message

call exit

max\_error:

print max\_error\_message

call exit

check\_array\_length endp

print\_array proc

print array\_elements\_message

mov cx, sizeArr

xor si, si

display\_loop:

mov ax, array[si]

push cx

push ax

print space

pop ax

call printNum

pop cx

add si, 2

loop display\_loop

ret

print\_array endp

printNum proc

mov bx, ax

or bx,bx

jns m1

mov al,'-'

int 29h

neg bx

m1:

xor cx ,cx

mov ax, bx

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

printNum endp

find\_max\_element proc

xor si,si

xor dx,dx

mov dx, array[si]

mov max\_element, dx ; element array[0]

mov cx, sizeArr

repeatl:

mov ax, array[si]

cmp max\_element, ax

jg no\_change

mov max\_element, ax

no\_change:

add si, 2

loop repeatl

print max\_element\_message

mov ax, max\_element

call printNum

ret

find\_max\_element endp

sort proc

push ax

push bx

push cx

push dx

push di

xor si,si

lea si, array

xor bx,bx

mov bx, sizeArr

mov ax, si

mov cx, bx

dec cx ;cx-1

outer\_loop:

mov bx,cx

mov si,ax

mov di,ax

add di,2 ; next elem

inner\_loop:

mov dx,[si]

cmp dx, [di]

jl skip\_swap

xchg dx,[di]

mov [si],dx

skip\_swap:

add si,2

add di,2

dec bx

jnz inner\_loop

loop outer\_loop

print sorted\_message

pop di

pop dx

pop cx

pop bx

pop ax

ret

sort endp

sum\_of\_array proc

.386

mov cx, sizeArr

xor ax, ax

xor bx, bx

xor si, si

m\_sum:

mov ax, array[si]

cwde

add ebx, eax

add si, 2

loop m\_sum

mov eax, ebx

print sum\_message

call printNum

ret

overflow1:

print overflow\_error

ret

sum\_of\_array endp

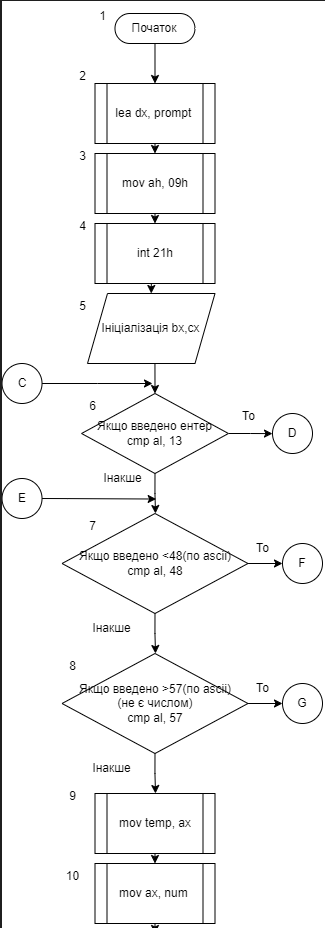
cseg ends

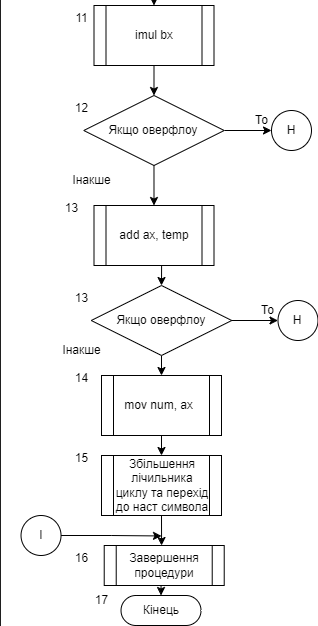
end main

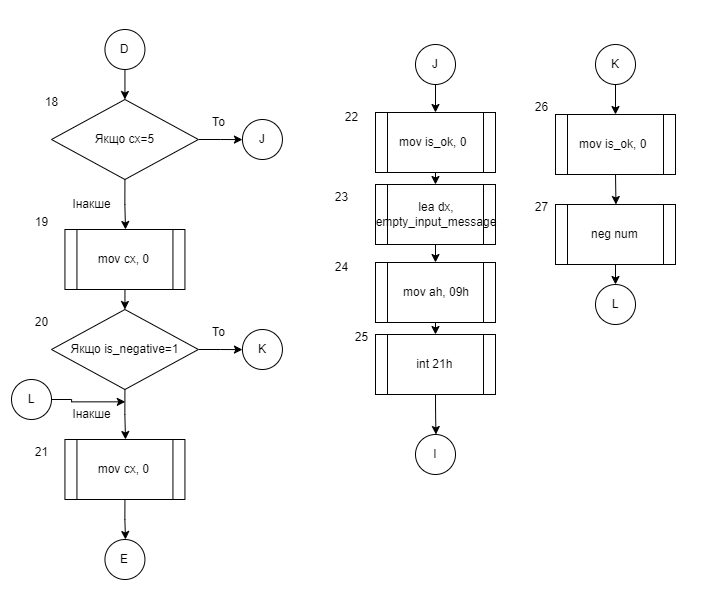
Блок-схема програми:

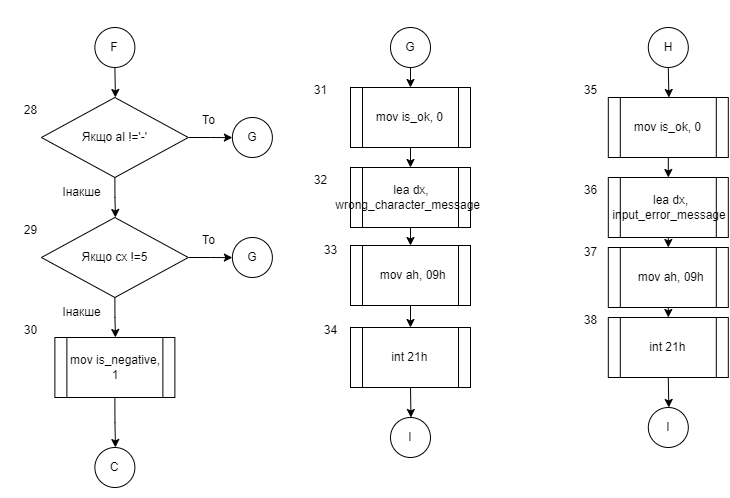


Блок-схема процедури read\_digit:

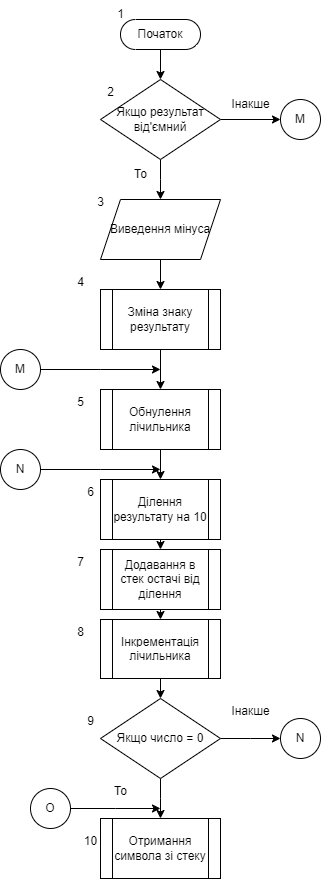


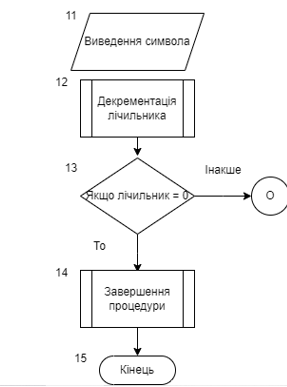




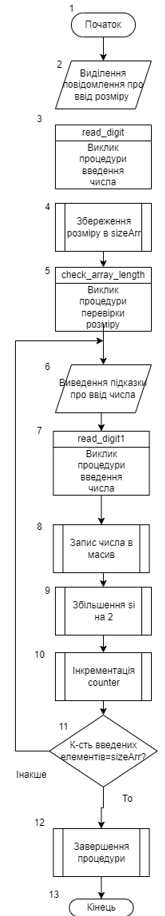


Блок-схема процедури printNum:

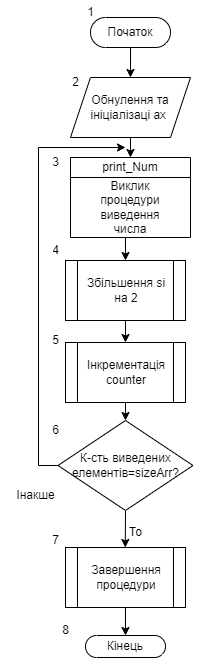




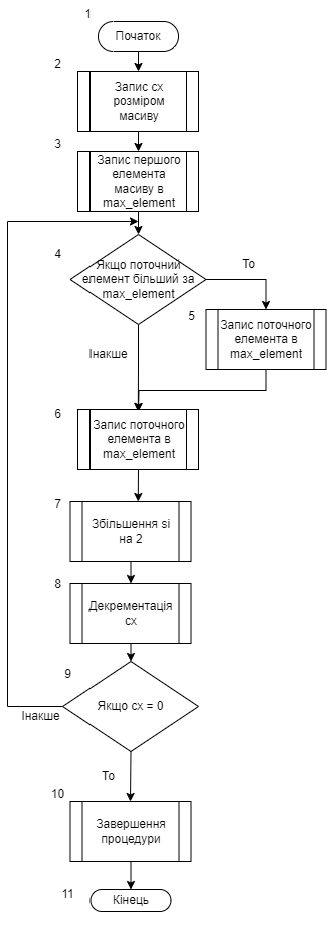
Блок-схема процедури enterArray:



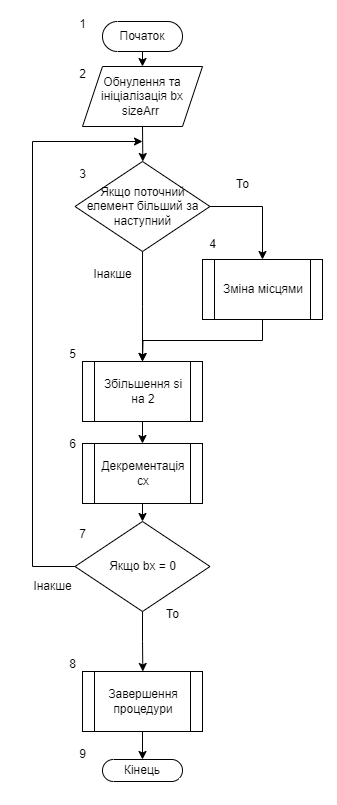
Блок-схема процедури PrintArray:



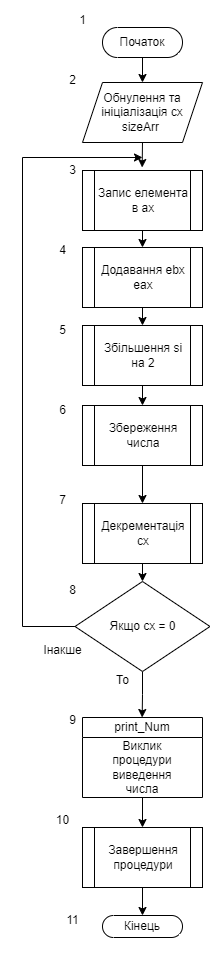
Блок-схема процедури find\_max\_element:

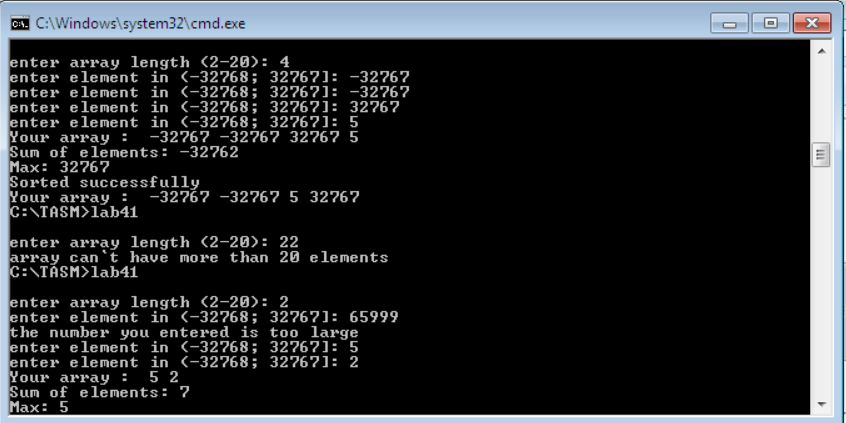


Блок-схема процедури sort:



Блок-схема процедури sum\_of\_Array:



**Тестування програми:**   
  
Висновок: В текстовому редакторі написав програму за своїм варіантом завдання в файлі .asm. Програма приймає від користувача розмір одномірного масиву та значення його елементів, знаходить суму, його максимальне число та сортує бульбашкою та виводить результат. Побудував блок-схеми програми. Програма має захист від некоректно введених даних  
Таким чином, програма коректно виконує поставлену задачу.