НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Звіти до комп’ютерних практикумів з кредитного модуля «Програмування, ч. III» «Системне програмування»

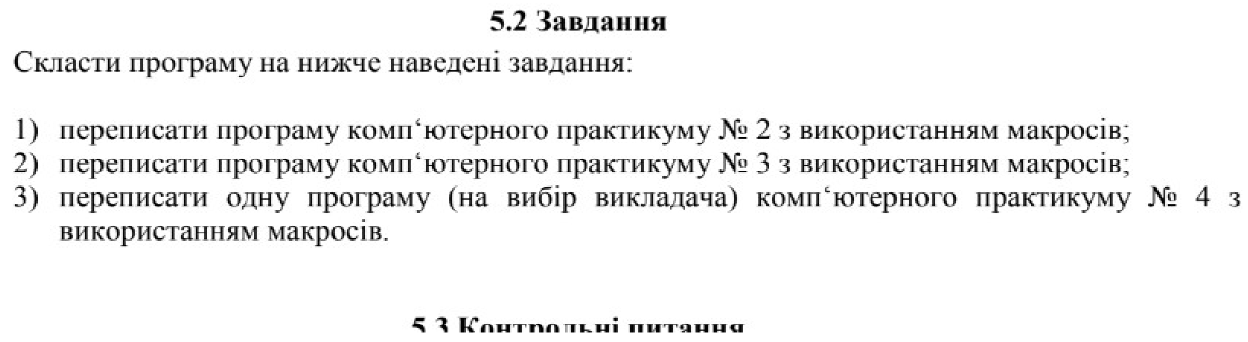
Прийняв	**Виконав**
доцент кафедри ТК	**Студент групи ІТ-02**
Лісовиченко О.І.	**Макаров І.С.**
29.12.2020р	

Київ – 2020

Комп’ютерний практикум No5

Тема: макрозасоби мови асемблер.

Завдання:



5.2 Завдання

Скласти програму на нижче наведені завдання:

1) переписати програму комп‘ютерного практикуму № 2 з використанням макросів;
2) переписати програму комп‘ютерного практикуму № 3 з використанням макросів;
3) переписати одну програму (на вибір викладача) комп‘ютерного практикуму № 4 з використанням макросів.

5.3 Контрольні питання

Приклад коду:

```
.model large


STK SEGMENT PARA STACK "STACK"
DB 64 dup ('STACK')
STK ENDS

DSEG SEGMENT PARA PUBLIC "DATA"
digit dw ? ; input X will be stored here

numerator dw ?
divider dw ?

resultFloat dw ?
resultInt dw ?

coutFloat dw ?

negativeArraySizeErrorMessage db 13, 10, 'Array size cannot be negative or 0$'
nanErrorMessage db 13, 10, 'You entered not a number$'
outOfRangeErrorMessage db 13, 10, 'Index out of range$'
invalidChoiceMessage db 13, 10, 'You entered invalid number, chose 1, 2, 3 or 4'

newLine db '', 0Dh, 0Ah, '$'

inputWhatTypeOfActionUserWantMessage1 db 13, 10, 'Enter 1 if you want to find sum $'
inputWhatTypeOfActionUserWantMessage2 db 13, 10, ' 2 for MIN and MAX value $'
inputWhatTypeOfActionUserWantMessage3 db 13, 10, ' 3 for sorting $'
inputWhatTypeOfActionUserWantMessage4 db 13, 10, ' 4 for finding coordinates: $'

inputArraySizeMessage db 13, 10, 'Enter the size of an array: $'
```

```
resultSumIsMessage db 13, 10, 'The sum of the array is: $'
resutlMinMaxMessage db 13, 10, 'Maximum and minimum values respectively are: $'
resultSortingMessage db 13, 10, 'Sorted array: $'
resultFindingIndexes db 13, 10, 'Here are indexes of target value: $'
inputMatrixFirstDimensionMessage db 13, 10, 'Enter the first dimension of an array: $'
inputMatrixSecondDimensionMessage db 13, 10, 'Enter the second dimension of an array: 
$'
inputValueToFind db 13, 10, 'Enter value you wanna find: $'


firstElement db 1 ; first element in row
isNegative dw 0 ; is our number negative
isError db 0
NUM db 7, ?, 7 dup('?')

userChoseSum db ?
userChoseMinMax db ?
userChoseSorting db ?
userChoseCoordinates db ?


arrayLengthForSum dw ?
arrayLengthForCout dw ?
arrayLengthForSortingForOuterCycle dw ?
arrayLengthForSortingForInnerCycle dw ?
arrayLengthForCin dw ?
arrayLength dw ?

startIndexForCout dw ?

matrixWidth db ?
matrixHeight db ?

arraySum dw 0

currentIndex dw ?

minValue dw ?
maxValue dw ?

swapIndex1 dw ?
swapIndex2 dw ?
swapBuffer1 dw ?
swapBuffer2 dw ?

sortingOuterLoopCounter dw ?
sortingInnerLoopCounter dw ?
sortingInnerLoopLimit dw ?
sortingFirstValue dw ?
sortingSecondValue dw ?
```

```
firstMatrixDimension dw ?
secondMatrixDimension dw ?
firstMatrixDimensionCopy dw ?
matrixSize dw ?

targetValueToFind dw ?
xCoordinateOfTarget dw ?
yCoordinateOfTarget dw ?

array dw 40 dup(?)
DSEG ENDS

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

COUT PROC near
mov bx, digit

CMP digit, 0
JGE 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
COUT ENDP

CIN PROC
lea DX, NUM
XOR AX, AX
mov ah, 10
```

```
int 21h

lea SI, NUM + 1
MOV CL, [SI]

LEA DI, NUM + 2
mov DX, 0
XOR SI, SI
start:
MOV BL, [DI]

A1:
CMP BL, '0'
JB MINUS
CMP BL, '9'
JA MINUS
JMP number

number:
MOV AX, DX
MOV bX, 10
MUL bX
jo ERROR_OUT_OF_RANGE
MOV DX, AX
XOR AX, AX
MOV AL, [DI]
SUB AX, 30h
ADD DX, AX
jo ERROR_OUT_OF_RANGE

CMP DX, 32769
JA ERROR_OUT_OF_RANGE

INC DI
xor BX, BX
mov firstElement, 0
LOOP start

mov digit, DX

CMP isNegative, 1
JE negotiate
JNE endcin

negotiate:
NEG digit
JMP endcin

MINUS:
XOR AX, AX
CMP firstElement, 1
```

```
JNE ERROR_NAN
CMP NUM + 2 , '-'
JE MINUSW
JMP ERROR_NAN
ERROR_NAN:
MOV AH, 09
LEA DX, nanErrorMessage
INT 21h
MOV isError, 1
JMP endcin
ERROR_OUT_OF_RANGE:
MOV AH, 09
LEA DX, outOfRangeErrorMessage
INT 21h
MOV isError, 1
JMP endcin
MINUSW:
MOV isNegative, 1 ; negative
MOV firstElement, 0 ; not first
INC DI
DEC CL
JMP START
endcin:
NEG digit
NEG digit
RET
CIN ENDP

printNewLine PROC
LEA DX, newLine
MOV AH, 09
int 21h
ret
printNewLine ENDP

checkWhatActionToDo PROC
CMP digit, 1
JE isFirst

CMP digit, 2
JE isSecond

CMP digit, 3
JE isThird

CMP digit, 4
JE isFourth

showCheckingError:
MOV isError, 1
LEA DX, invalidChoiceMessage
```

```
MOV AH, 09h
INT 21h

JMP endChecking
isFirst:
MOV userChoseSum, 1
JMP endChecking
isSecond:
MOV userChoseMinMax, 1
JMP endChecking
isThird:
MOV userChoseSorting, 1
JMP endChecking
isFourth:
MOV userChoseCoordinates, 1
JMP endChecking
endChecking:
RET
checkWhatActionToDo ENDP

printAskingForAction PROC
LEA DX, inputWhatTypeOfActionUserWantMessage1
MOV AH, 09h
INT 21h
CALL printNewLine

LEA DX, inputWhatTypeOfActionUserWantMessage2
MOV AH, 09h
INT 21h
CALL printNewLine

LEA DX, inputWhatTypeOfActionUserWantMessage3
MOV AH, 09h
INT 21h
CALL printNewLine

LEA DX, inputWhatTypeOfActionUserWantMessage4
MOV AH, 09h
INT 21h

RET
printAskingForAction ENDP

inputArraySize PROC
LEA DX, inputArraySizeMessage
MOV AH, 09h
INT 21h

CALL CIN
CALL printNewLine
```

```
CMP digit, 0
JLE arraySizeNegative

MOV AX, digit

MOV arrayLength, AX
MOV arrayLengthForCout, AX
MOV arrayLengthForSortingForOuterCycle, AX
MOV arrayLengthForSortingForInnerCycle, AX
MOV arrayLengthForSum, AX
MOV arrayLengthForCin, AX

endInputArraySize:
RET

arraySizeNegative:
MOV isError, 1
LEA DX, arraySizeNegative
MOV AH, 09h
INT 21h

JMP endInputArraySize
inputArraySize ENDP

cinArray PROC
XOR SI, SI
MOV currentIndex, 0
cinElement:
MOV firstElement, 1
MOV isNegative, 0
CALL CIN
CALL printNewLine

CMP isError, 1
JE endArrayCin

MOV SI, currentIndex
MOV AX, digit
MOV array[SI], AX

ADD SI, 2
MOV currentIndex, SI

MOV CX, arrayLengthForCin
DEC CX
MOV arrayLengthForCin, CX

CMP CX, 0
JG cinElement
endArrayCin:
RET
```

```
cinArray ENDP

coutArray PROC
MOV AL, '['
INT 29h
MOV AL, ' '
INT 29h

XOR SI, SI
MOV SI, startIndexForCout

coutElements:
MOV AX, array[SI]
MOV digit, AX

CALL COUT

ADD SI, 2

MOV AL, ' '
INT 29h

MOV startIndexForCout, SI

MOV CX, arrayLengthForCout
DEC CX
MOV arrayLengthForCout, CX

CMP CX, 0
JG coutElements
MOV AL, ']'
INT 29h

RET
coutArray ENDP

calculateSum PROC
XOR SI, SI
MOV SI, 0
MOV AX, 0
MOV arraySum, AX

MOV CX, arrayLengthForSum

startCalculatingSum:
MOV AX, array[SI]
ADD SI, 2

ADD arraySum, AX
LOOP startCalculatingSum
RET
```

```
calculateSum ENDP

sumChoice PROC
CALL calculateSum
LEA DX, resultSumIsMessage
MOV AH, 09h
INT 21h

MOV AX, arraySum
MOV digit, AX
CALL COUT

RET
sumChoice ENDP

minMaxChoice PROC
XOR SI, SI
MOV SI, 0

MOV AX, array[SI]
MOV minValue, AX
MOV maxValue, AX

MOV CX, arrayLength
startFindingMinMaxValue:
MOV AX, array[SI]

CMP AX, minValue
JLE setMinValue
CMP AX, maxValue
JGE setMaxValue

JMP continueFindingMinMax

setMinValue:
MOV minValue, AX
JMP continueFindingMinMax
setMaxValue:
MOV maxValue, AX
JMP continueFindingMinMax

continueFindingMinMax:
ADD SI, 2
LOOP startFindingMinMaxValue
LEA DX, resutlMinMaxMessage
MOV AH, 09h
INT 21h

MOV AX, maxValue
MOV digit, AX
CALL COUT
```

```
MOV AL, ' '
INT 29h

MOV AX, minValue
MOV digit, AX
CALL COUT

RET
minMaxChoice ENDP

swapArrayElements PROC
MOV SI, swapIndex1
MOV AX, array[SI]

MOV swapBuffer1, AX

MOV SI, swapIndex2
MOV AX, array[SI]

MOV swapBuffer2, AX

MOV SI, swapIndex1
MOV AX, swapBuffer2
MOV array[SI], AX

MOV SI, swapIndex2
MOV AX, swapBuffer1
MOV array[SI], AX

RET
swapArrayElements ENDP

bubbleSort PROC
MOV sortingOuterLoopCounter, 0
outerLoop:

MOV AX, arrayLength
SUB AX, 1
MOV sortingInnerLoopLimit, AX

MOV sortingInnerLoopCounter, 0
innerLoop:
MOV AX, sortingInnerLoopCounter
MOV BX, 2
MUL BX
MOV SI, AX
MOV swapIndex1, AX

MOV AX, array[SI]
MOV sortingFirstValue, AX
```

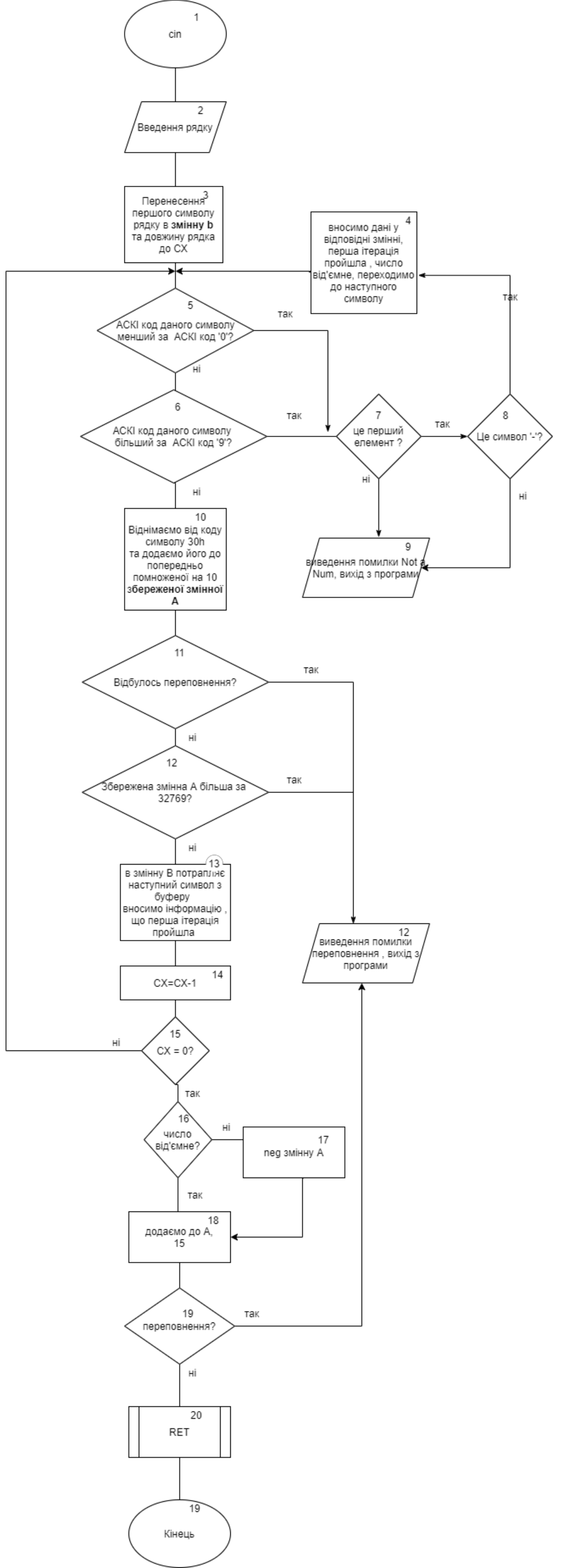
```
ADD SI, 2
MOV swapIndex2, SI
MOV AX, array[SI]
MOV sortingSecondValue, AX

MOV AX, sortingSecondValue
CMP sortingFirstValue, AX
JG performSwapPoint

JMP continueSorting

performSwapPoint:
CALL swapArrayElements

continueSorting:
MOV AX, sortingInnerLoopCounter
INC AX
MOV sortingInnerLoopCounter, AX

MOV AX, sortingInnerLoopLimit
CMP sortingInnerLoopCounter, AX
JL innerLoop

MOV CX, arrayLengthForSortingForOuterCycle
DEC CX
MOV arrayLengthForSortingForOuterCycle, CX
CMP CX, 0
JG outerLoop

RET
bubbleSort ENDP

sortingChoice PROC
CALL bubbleSort
LEA DX, resultSortingMessage
MOV AH, 09h
INT 21h

MOV startIndexForCout, 0
MOV AX, arrayLength
MOV arrayLengthForCout, AX
CALL coutArray

RET
sortingChoice ENDP

coutMatrix PROC
MOV currentIndex, 0
outerLoopForCouting:
MOV AX, currentIndex
MOV BX, 2
MUL BX
MOV BX, secondMatrixDimension
MUL BX ; now in AX laying value of current row index

MOV startIndexForCout, AX

MOV AX, secondMatrixDimension
MOV arrayLengthForCout, AX
CALL coutArray
CALL printNewLine

MOV CX, firstMatrixDimensionCopy
DEC CX
MOV firstMatrixDimensionCopy, CX

MOV AX, currentIndex
INC AX
MOV currentIndex, AX

CMP CX, 0
JG outerLoopForCouting

RET
coutMatrix ENDP

printCoordinates PROC
MOV AL, '['
INT 29h

MOV AX, yCoordinateOfTarget
MOV digit, AX
CALL COUT

MOV AL, ','
INT 29h
MOV AL, ' '
INT 29h

MOV AX, xCoordinateOfTarget
MOV digit, AX
CALL COUT

MOV AL, ']'
INT 29h

MOV AL, ' '
INT 29h

RET
printCoordinates ENDP
```

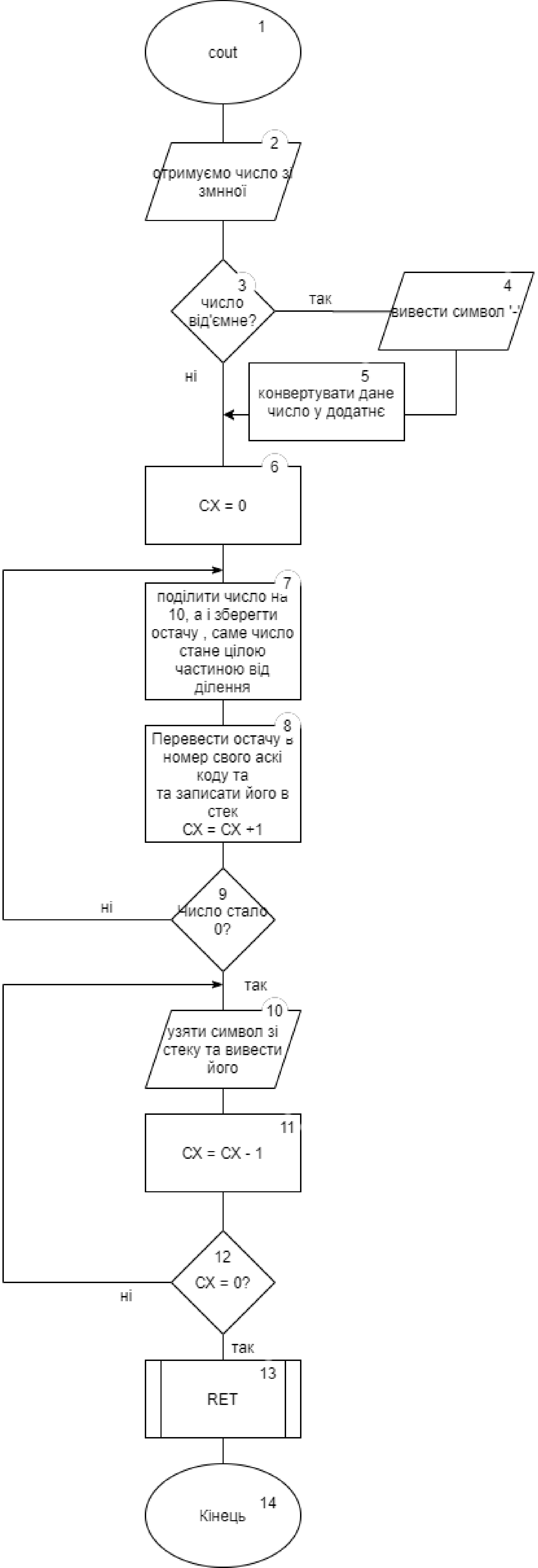
```
findTarget PROC
MOV yCoordinateOfTarget, 0
outerLoopOfSearching:
MOV AX, yCoordinateOfTarget
MOV BX, 2
MUL BX
MOV BX, secondMatrixDimension
MUL BX ; now in AX laying value of current row index

MOV SI, AX

MOV xCoordinateOfTarget, 0
innerLoopOfSearching:
MOV AX, array[SI]
ADD SI, 2

CMP targetValueToFind, AX
JNE continueFindingValue

CALL printCoordinates

continueFindingValue:
ADD xCoordinateOfTarget, 1
MOV AX, secondMatrixDimension

CMP xCoordinateOfTarget, AX
JL innerLoopOfSearching



ADD yCoordinateOfTarget, 1
MOV AX, firstMatrixDimension

CMP yCoordinateOfTarget, AX
JL outerLoopOfSearching
RET
findTarget ENDP

coordinatesChoice PROC
LEA DX, inputMatrixFirstDimensionMessage
MOV AH, 09h
INT 21h

CALL CIN
MOV AX, digit
MOV firstMatrixDimension, AX
MOV firstMatrixDimensionCopy, AX

CALL printNewLine
```

```
LEA DX, inputMatrixSecondDimensionMessage
MOV AH, 09h
INT 21h

CALL CIN
CALL printNewLine

MOV AX, digit
MOV secondMatrixDimension, AX
MOV BX, firstMatrixDimension
MUL BX ; now in AX there is matrix size = first_dim * second_dim

MOV matrixSize, AX
MOV arrayLengthForCin, AX

CALL cinArray
CALL printNewLine

CALL coutMatrix
LEA DX, inputValueToFind
MOV AH, 09h
INT 21h

MOV firstElement, 1
MOV isNegative, 0
CALL CIN

MOV AX, digit
MOV targetValueToFind, AX

CALL printNewLine

LEA DX, resultFindingIndexes
MOV AH, 09h
INT 21h

CALL findTarget
RET
coordinatesChoice ENDP

initializeArray PROC
CALL inputArraySize
CMP isError, 1
JE endInit

CALL cinArray
CMP isError, 1
JE endInit

MOV startIndexForCout, 0
CALL coutArray
CALL printNewLine

endInit:
RET
initializeArray ENDP

MAIN PROC
PUSH DS
MOV AX, 0
PUSH AX
MOV AX, DSEG
MOV DS, AX

CALL printAskingForAction
CALL CIN
CALL printNewLine
CMP isError, 1
JNE next1
JMP endF

next1:

CALL checkWhatActionToDo
CALL printNewLine
CMP isError, 1
JE endF

CMP userChoseSum, 1
JE findSumPoint
CMP userChoseMinMax, 1
JE findMinMaxPoint
CMP userChoseSorting, 1
JE sortArrayPoint
CMP userChoseCoordinates, 1
JE findCoordinatesPoint

findSumPoint:
CALL initializeArray

CMP isError, 1
JE endF

CALL sumChoice
JMP endF
findMinMaxPoint:
CALL initializeArray
CMP isError, 1
JE endF

CALL minMaxChoice
JMP endF
```

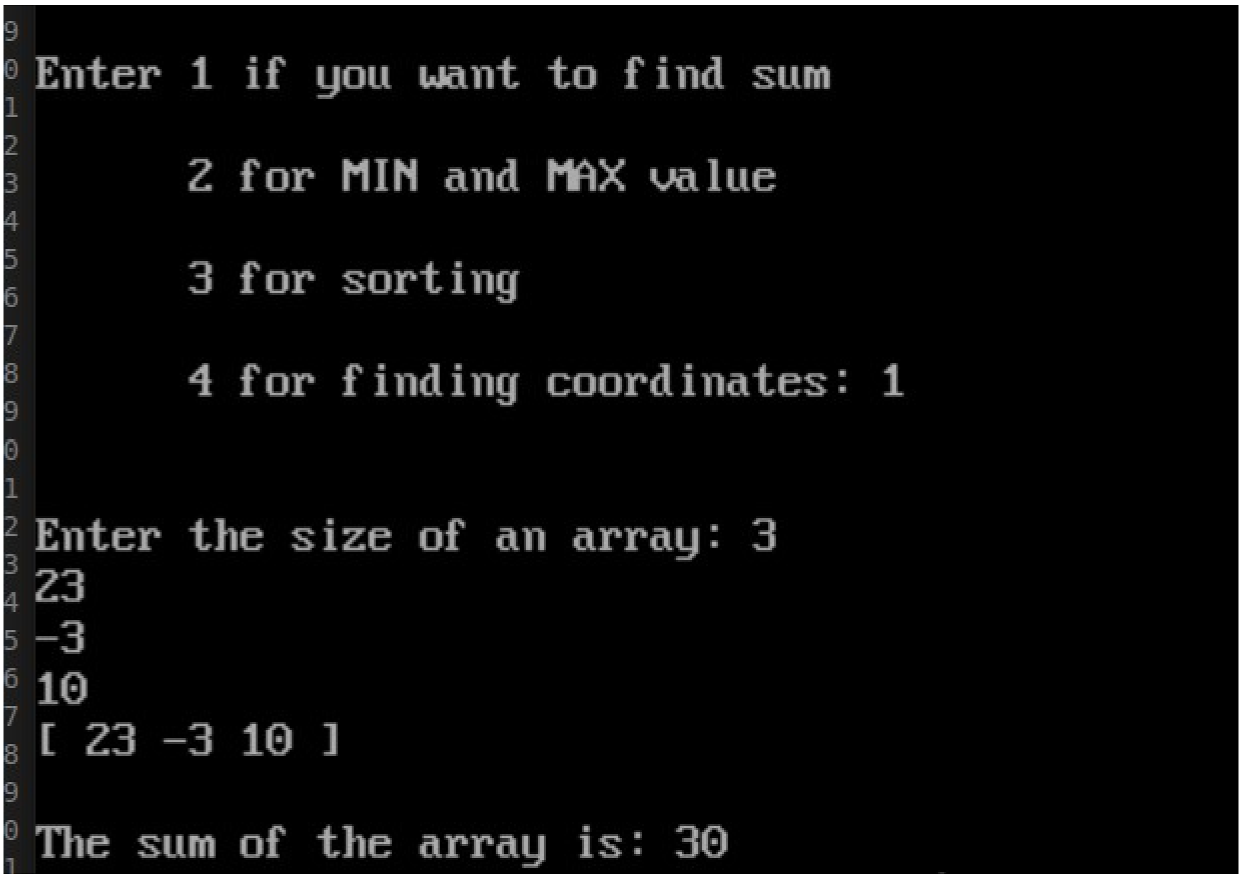
```
sortArrayPoint:
CALL initializeArray

CMP isError, 1
JE endF

CALL sortingChoice
JMP endF
findCoordinatesPoint:
CALL coordinatesChoice
JMP endF

endF:
RET
MAIN ENDP
CSEG ENDS

END MAIN
```

Схема функціонування програми

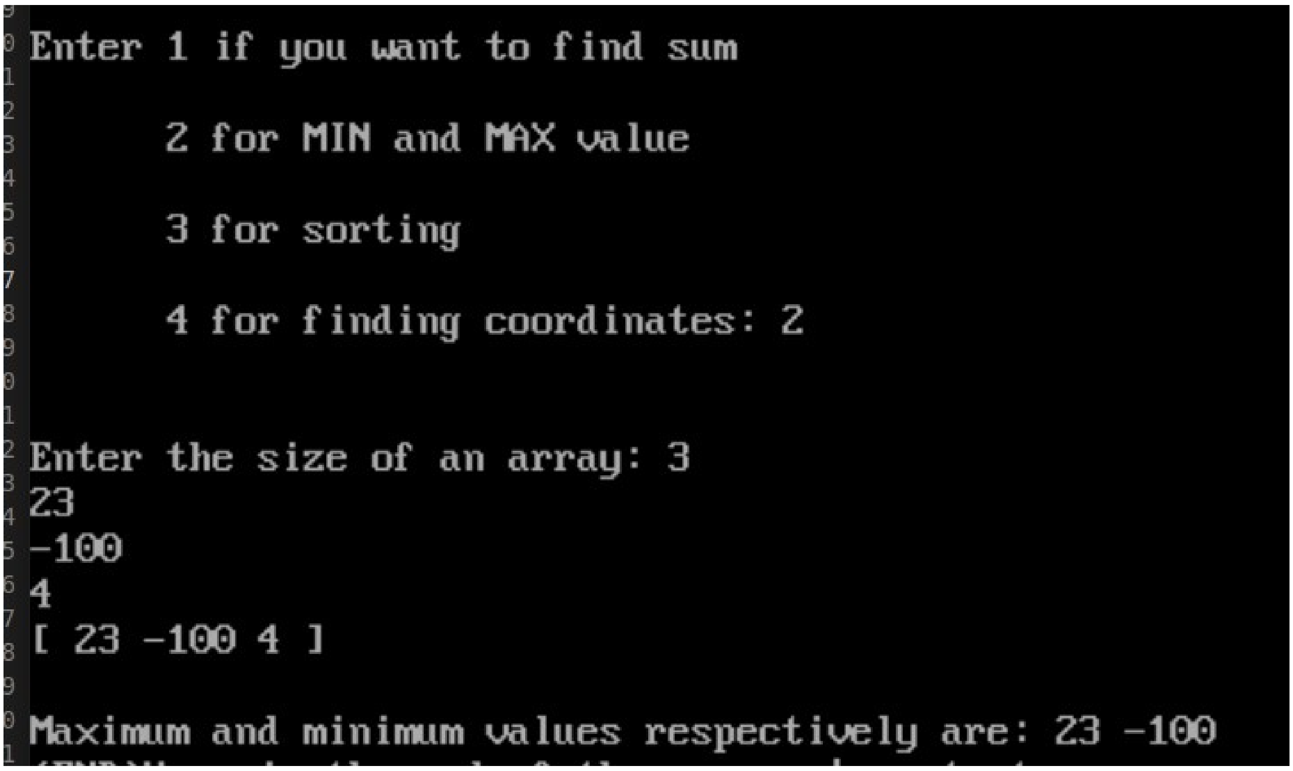

1 cin
2 Введення рядку
3 Перенесення першого символу рядку в змінну b та довжину рядка до CX
4 вносимо дані у відповідні змінні, перша ітерація пройшла , число від'ємне, переходимо до наступного символу
так
5 АСКІ код даного символу менший за АСКІ код '0'?
так
ні
6 АСКІ код даного символу більший за АСКІ код '9'?
так
7 це перший елемент ?
так
8 Це символ '-'?
ні
ні
9 виведення помилки Not a Num, вихід з програми
ні
10 Віднімаємо від коду символу 30h та додаємо його до попередньо помноженої на 10 збереженої змінної A
11 Відбулось переповнення?
так
ні
12 Збережена змінна A більша за 32769?
так
ні
13 в змінну B потрапляє наступний символ з буферу вносимо інформацію , що перша ітерація пройшла
12 виведення помилки переповнення , вихід з програми
14 CX=CX-1
15 CX = 0?
ні
так
16 число від'ємне?
ні
17 neg змінну A
так
18 додаємо до A, 15
19 переповнення?
так
ні
20 RET
19 Кінець


1
cout
2
отримуємо число зі змнної
3
число від'ємне?
так
4
вивести символ '-'
ні
5
конвертувати дане число у додатнє
6
CX = 0
7
поділити число на 10, а і зберегти остачу , саме число стане цілою частиною від ділення
8
Перевести остачу в номер свого аскі коду та та записати його в стек
CX = CX +1
9
Число стало 0?
ні
так
10
узяти символ зі стеку та вивести його
11
CX = CX - 1
12
CX = 0?
ні
так
13
RET
14
Кінець

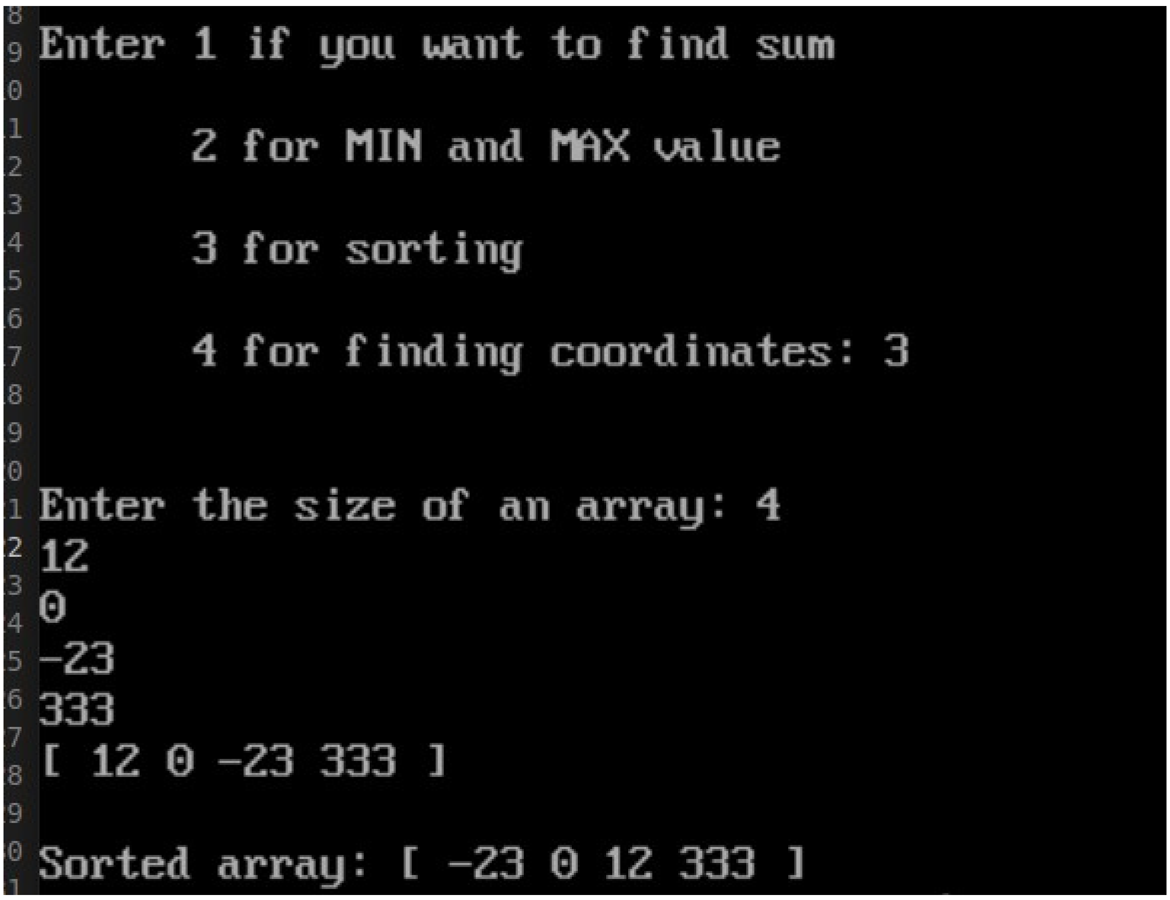
Результат роботи:



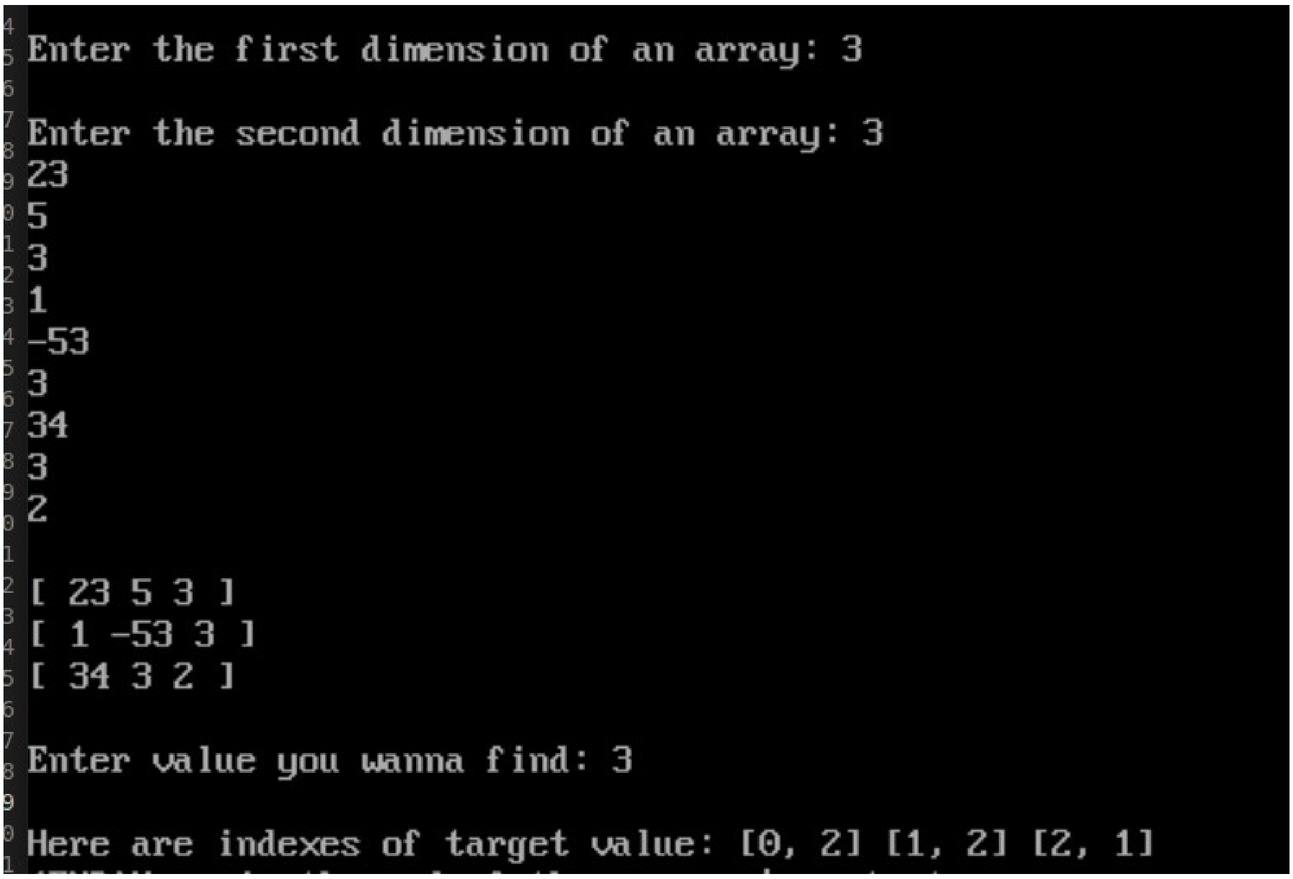
```
Enter 1 if you want to find sum

      2 for MIN and MAX value

      3 for sorting

      4 for finding coordinates: 1


Enter the size of an array: 3
23
-3
10
[ 23 -3 10 ]

The sum of the array is: 30
```



```
Enter 1 if you want to find sum

      2 for MIN and MAX value

      3 for sorting

      4 for finding coordinates: 2


Enter the size of an array: 3
23
-100
4
[ 23 -100 4 ]

Maximum and minimum values respectively are: 23 -100
```



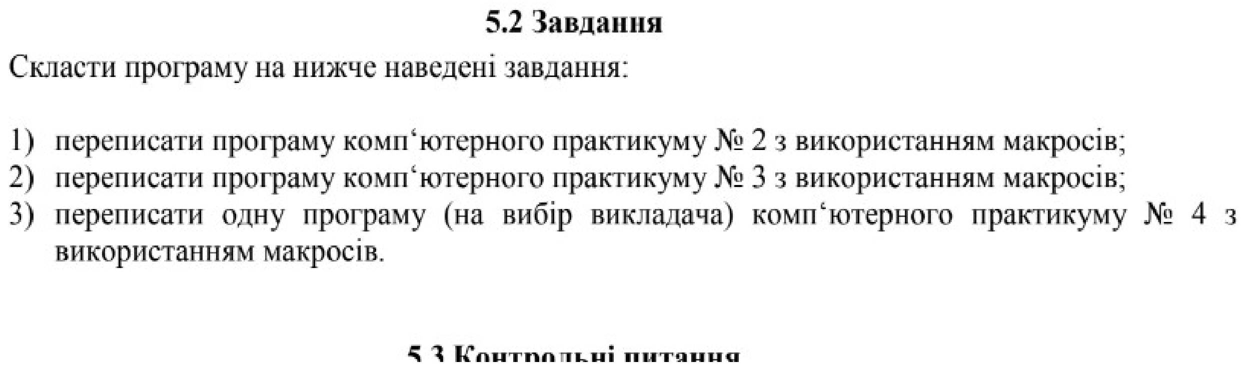
```
Enter 1 if you want to find sum

       2 for MIN and MAX value

       3 for sorting

       4 for finding coordinates: 3


Enter the size of an array: 4
12
0
-23
333
[ 12 0 -23 333 ]

Sorted array: [ -23 0 12 333 ]
```



```
Enter the first dimension of an array: 3

Enter the second dimension of an array: 3
23
5
3
1
-53
3
34
3
2

[ 23 5 3 ]
[ 1 -53 3 ]
[ 34 3 2 ]

Enter value you wanna find: 3

Here are indexes of target value: [0, 2] [1, 2] [2, 1]
```

Висновок:

1. Написав програму , для завдання



5.2 Завдання

Скласти програму на нижче наведені завдання:

1) переписати програму комп‘ютерного практикуму № 2 з використанням макросів;
2) переписати програму комп‘ютерного практикуму № 3 з використанням макросів;
3) переписати одну програму (на вибір викладача) комп‘ютерного практикуму № 4 з використанням макросів.

5.3 Контрольні питання

2. Програма передбачає введення даних , що не зможе обчислити система