

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

доцент кафедри ТК

Лісовиченко О.І.

29.12.2020р

Виконав

Студент групи ІТ-02

Макаров І.С.

Комп'ютерний практикум №4

Тема: програмування розгалужених алгоритмів

Завдання:

Скласти програму на нижче наведені завдання:

1. Написати програму знаходження суми елементів одномірного масиву, елементи вводить користувач.
2. Написати програму пошуку максимального (або мінімального) елемента одномірного масиву, елементи вводить користувач.
3. Написати програму сортування одномірного масиву цілих чисел загального вигляду.
4. Написати програму пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий вводить користувач.

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: \$'
resultMinMaxMessage 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, '-1'
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 minVal, AX
```

```
MOV maxVal, AX
```

```
MOV CX, arrayLength
```

```
startFindingMinMaxValue:
```

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

```
CMP AX, minVal
```

```
JLE setMinVal
```

```
CMP AX, maxVal
```

```
JGE setMaxVal
```

```
JMP continueFindingMinMax
```

```
setMinVal:
```

```
MOV minVal, AX
```

```
JMP continueFindingMinMax
```

```
setMaxVal:
```

```
MOV maxVal, AX
```

```
JMP continueFindingMinMax
```

```
continueFindingMinMax:
```

```
ADD SI, 2
```

```
LOOP startFindingMinMaxValue
```

```
LEA DX, resultMinMaxMessage
```

```
MOV AH, 09h
```

```
INT 21h
```

```
MOV AX, maxVal
```

```
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

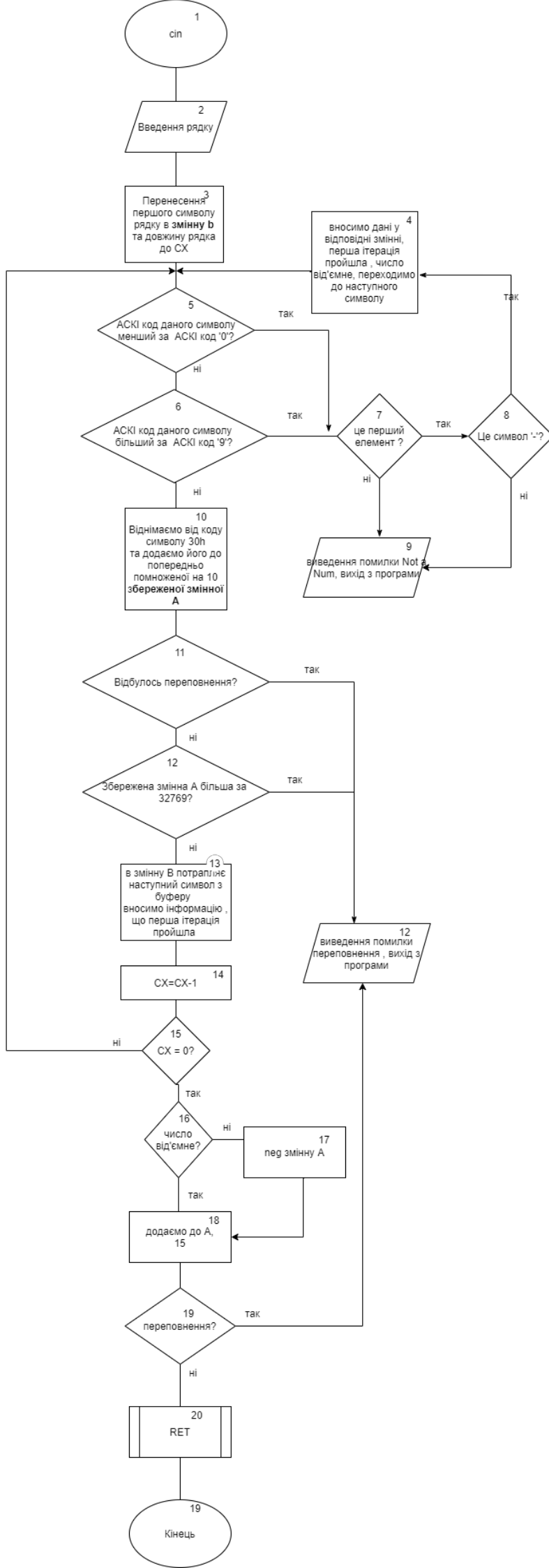
CMP isError, 1
JE endF

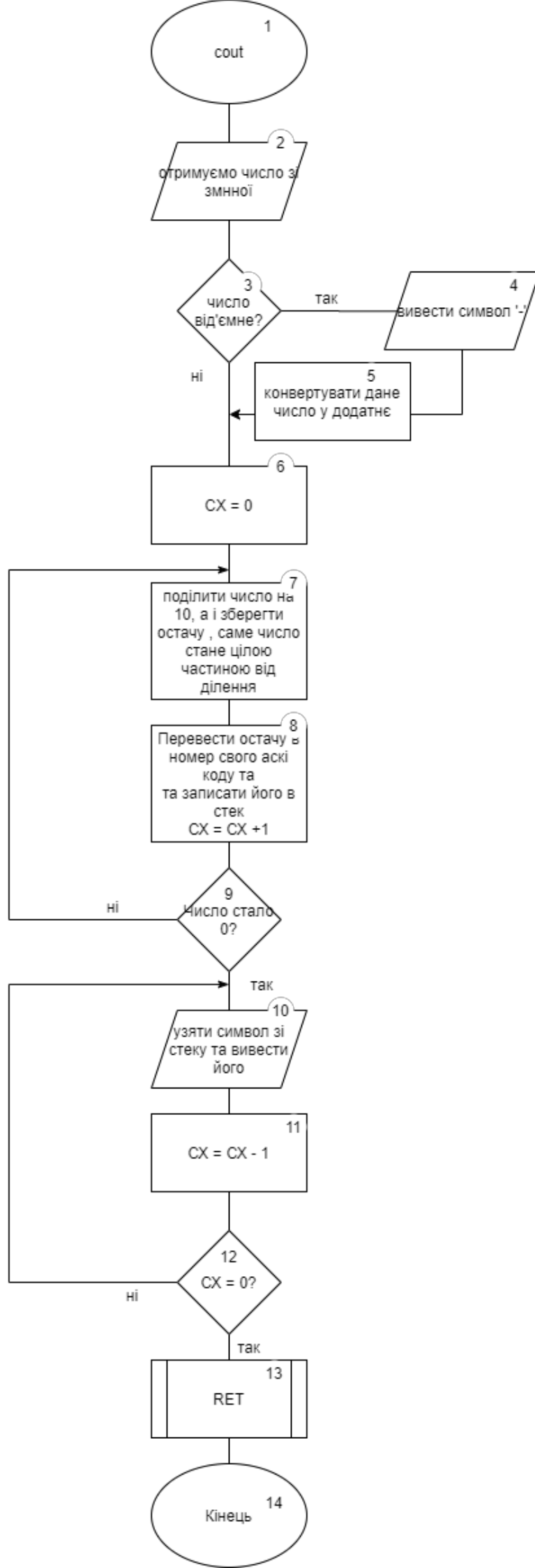
CALL sortingChoice
JMP endF
findCoordinatesPoint:
CALL coordinatesChoice
JMP endF

endF:
RET
MAIN ENDP
CSEG ENDS

END MAIN
```

Схема функціонування програми





Результат роботи:

```
9
0 Enter 1 if you want to find sum
1
2     2 for MIN and MAX value
3
4     3 for sorting
5
6     4 for finding coordinates: 1
7
8
9
0
1
2 Enter the size of an array: 3
3 23
4 -3
5 10
6 [ 23 -3 10 ]
7
8
9
0 The sum of the array is: 30
1
```

```
9
0 Enter 1 if you want to find sum
1
2     2 for MIN and MAX value
3
4     3 for sorting
5
6     4 for finding coordinates: 2
7
8
9
0
1
2 Enter the size of an array: 3
3 23
4 -100
5 4
6 [ 23 -100 4 ]
7
8
9
0 Maximum and minimum values respectively are: 23 -100
1
```

```

8 Enter 1 if you want to find sum
9
10
11     2 for MIN and MAX value
12
13     3 for sorting
14
15     4 for finding coordinates: 3
16
17
18
19
20
21 Enter the size of an array: 4
22 12
23 0
24 -23
25 333
26
27 [ 12 0 -23 333 ]
28
29
30 Sorted array: [ -23 0 12 333 ]
31

```

```

4 Enter the first dimension of an array: 3
5
6 Enter the second dimension of an array: 3
7
8 23
9
10 5
11
12 3
13
14 1
15
16 -53
17
18 3
19
20 34
21
22 3
23
24 2
25
26 [ 23 5 3 ]
27 [ 1 -53 3 ]
28 [ 34 3 2 ]
29
30 Enter value you wanna find: 3
31
32 Here are indexes of target value: [0, 2] [1, 2] [2, 1]
33

```

Висновок:

1. Написав програму , для завдання

Скласти програму на нижче наведені завдання:

1. Написати програму знаходження суми елементів одномірного масиву, елементи вводить користувач.
2. Написати програму пошуку максимального (або мінімального) елемента одномірного масиву, елементи вводить користувач.
3. Написати програму сортування одномірного масиву цілих чисел загального вигляду.
4. Написати програму пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий вводить користувач.

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