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 №4

Тема: масиви.

Завдання:

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 1:

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

DSEG SEGMENT PARA PUBLIC "DATA"
arr_size dw 0
inmes 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
 call input_digit
 call str_to_int
 cmp is_error, 1
 je raise_error_1
 cmp is_error, 2
 je raise_error_2
 cmp num, 0
 jle raise_error_2
 cmp num, 32
 jg raise_error_2
 mov ax, num
 mov arr_size, ax
 call array_input        ;entering array elements
 cmp is_error, 1
 je raise_error_1
 cmp is_error, 2
 je raise_error_2
 call array_sum          ;calculate and print the sum
 call result_print_dword
 call array_min_and_max  ;find and print min and max element
 lea dx, min_print
 mov ah, 9
 int 21h
 mov ax, min
 mov num, ax
 call result_print
 lea dx, max_print
 mov ah, 9
 int 21h
 mov ax, max
 mov num, ax
 call result_print
 call array_sort         ;sort array and print it
 call array_print
 jmp end_program
```

```
raise_error_1:
 LEA dx, error_message_1
 MOV ah,9
 INT 21h
 jmp end_program
raise_error_2:
 LEA dx, error_message_2
 MOV ah,9
 INT 21h
end_program:
 mov AH, 4CH
 int 21H
 ret
main endp

input_digit proc
 mov ah, 9              ;showing a tip
 int 21h
 lea dx, inmes          ;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
 ret
input_digit endp

str_to_int proc
 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
 ret
str_to_int endp

result_print proc
 mov bx, num
 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
result_print endp

result_print_dword proc
.386
 mov ebx, sum
 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
 ret
result_print_dword endp

array_input proc
 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
element_input:               ;input loop
 lea dx, el_tip              ;formatted input array element
 mov ah, 9
 int 21h
 mov ax, counter
 mov num, ax
 call result_print
 lea dx, el_input_tip
 call input_digit
 call str_to_int
 cmp is_error, 1
 je end_input
 cmp is_error, 2
 je end_input
 mov ax, num
 mov [arr+di], ax
 inc counter
 add di, 2
 mov cx, counter
 cmp cx, arr_size
 jne element_input
end_input:
 ret
```

```
array_input endp

array_sum proc
.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 sum, edx
 ret
array_sum endp

array_min_and_max proc
 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:
 ret
array_min_and_max endp

array_sort proc
 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:
 ret
array_sort endp

array_print proc
 lea dx, sorted_arr_print
 mov ah, 09h
 int 21h
 mov di, 0
 mov si, 0
show:
 mov dx, [arr+si]
 mov num, dx
 call result_print
```

```
 mov al, ' '
 int 29h
 add si, 2
 inc di
 cmp di, arr_size
 jne show
 ret
array_print endp

CSEG ENDS
end main
```

Текст програми 2:

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

DSEG SEGMENT PARA PUBLIC "DATA"
n dw 0        ;rows
m dw 0        ;columns
inmes db 7,?,7 dup (" $")
matr_tip db "Element [$"
matr_i_tip db "][$"
matr_ij_tip db "] => $"
is_negative db 0
num dw 0
digit dw 0
is_error db 0
error_message_1 db "Invalid symbol(s)!$"
error_message_2 db "Number is out of range!$"
n_input_tip db "Ammount of rows (1-16) => $"
m_input_tip db "Ammount of columns (1-16) => $"
el_input_tip db "Enter values in [-32768; 32767]:$"
arr dw 16 dup (16 dup (?))
i dw 0
j dw 0
value_entr_tip db "Enter seeken value => $"
no_entries_found db "There are no entries!$"
con_query db "Find any other value? ('y'-continue, else - stop) => $"
is_in_matrix db 0
tmp 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, n_input_tip                ;input row number
 call input_digit
 call str_to_int
 cmp is_error, 1
 je raise_error_1
 cmp is_error, 2
 je raise_error_2
 cmp num, 0
 jle raise_error_2
 cmp num, 16
 jg raise_error_2
 mov ax, num
 mov n, ax
 lea dx, m_input_tip                ;input column number
 call input_digit
 call str_to_int
 cmp is_error, 1
 je raise_error_1
 cmp is_error, 2
 je raise_error_2
 cmp num, 0
 jle raise_error_2
 cmp num, 16
 jg raise_error_2
 mov ax, num
 mov m, ax
 call matr_input                    ;entering matrix elements
 cmp is_error, 1
 je raise_error_1
 cmp is_error, 2
 je raise_error_2
 call find_entries_dialog           ;seeking of elment coordinates
 jmp end_program
raise_error_1:
 LEA dx, error_message_1
 MOV ah,9
 INT 21h
 jmp end_program
raise_error_2:
 LEA dx, error_message_2
 MOV ah,9
```

```
 INT 21h
end_program:
 mov AH, 4CH
 int 21H
 ret
main endp

input_digit proc
 mov ah, 9              ;showing a tip
 int 21h
 lea dx, inmes          ;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
 ret
input_digit endp

str_to_int proc
 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
 jne done
 neg num
done:
 xor ch, ch
 ret
str_to_int endp

result_print proc
 mov bx, num
 or bx, bx
 jns m1
```

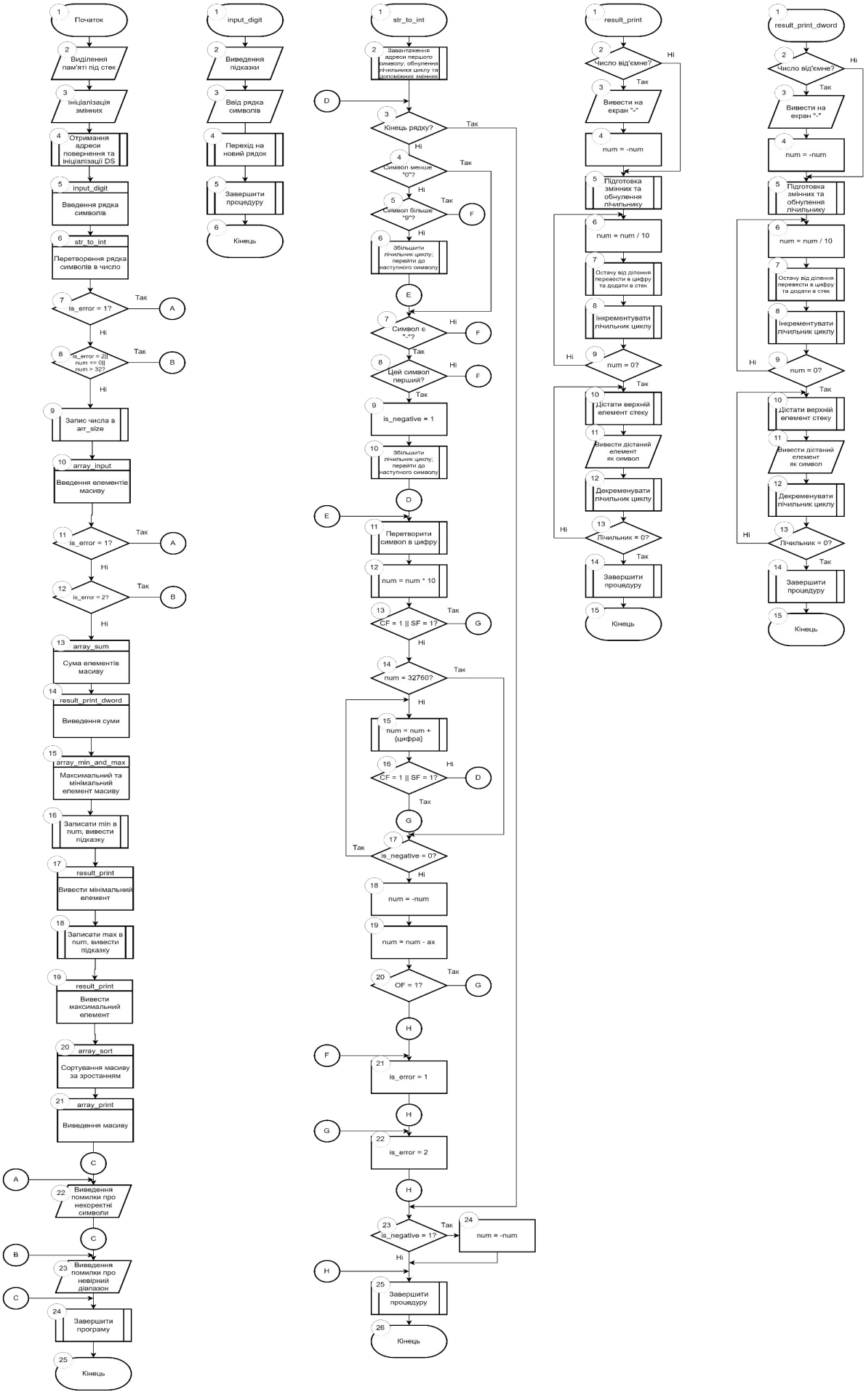
```
 mov al, '-'
 int 29h
 neg bx
m1:
 mov ax, bx
 xor cx, cx
 mov bx, 10
m2:
 xor dx, dx
 div bx
 add dl, '0'
 push dx
 inc cx
 test ax, ax
 jnz m2
m3:
 pop ax
 int 29h
 loop m3
 ret
result_print endp

matr_input proc
 lea dx, el_input_tip          ;print a tip
 mov ah, 9
 int 21h
 mov al,10                     ;print a new line after the input
 int 29h
 mov al,13
 int 29h
 mov di, 0
element_input:                 ;input loop
 lea dx, matr_tip              ;formatted input and convert matrix element
 mov ah, 9
 int 21h
 mov ax, i
 inc ax
 mov num, ax
 call result_print
 mov num, 0
 lea dx, matr_i_tip
 mov ah, 9
 int 21h
 mov ax, j
 inc ax
```

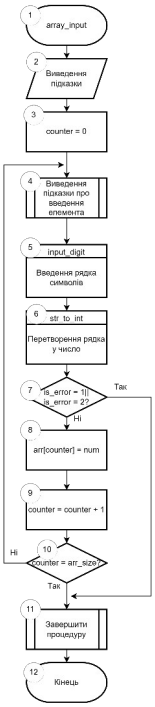
```
 mov num, ax
 call result_print
 mov num, 0
 lea dx, matr_ij_tip
 call input_digit
 call str_to_int
 cmp is_error, 1
 je end_input
 cmp is_error, 2
 je end_input
 mov ax, num
 mov [arr+di], ax
 add di, 2
 inc j
 mov cx, j                    ;if end of row, go to next row
 cmp cx, m
 jl element_input
 mov j, 0
 inc i
 mov bx, i
 mov ax, 32
 mul bl
 mov di, ax
 cmp bx, n                    ;if end of last row, end input
 jl element_input
end_input:
 ret
matr_input endp

find_entries_dialog proc
dialog_loop:
 mov is_error, 0
 mov is_in_matrix, 0
 lea dx, value_entr_tip
 call input_digit
 call str_to_int
 cmp is_error, 1
 je no_such_values
 cmp is_error, 2
 je no_such_values
 call find_and_print_entries          ;procedure to find and print entered value's
entries
 cmp is_in_matrix, 0                  ;print that no entries of entered value if it wasn`t
found
 je no_such_values
```

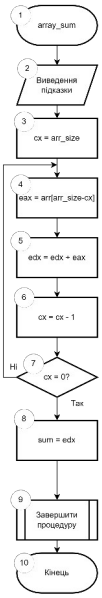
```
continue_dialog:
 lea dx, con_query                    ;ask user if he wants to continue searching for
values' entries
 call input_digit
 mov dl, [inmes+2]
 cmp dl, 'y'
 je dialog_loop
 jne end_proc
no_such_values:
 lea dx, no_entries_found
 mov ah, 9
 int 21h
 mov al,10                           ;print a new line after the input
 int 29h
 mov al,13
 int 29h
 jmp continue_dialog
end_proc:
 ret
find_entries_dialog endp

find_and_print_entries proc
 mov ax, num
 mov tmp, ax
find_entries:
 mov i, 0                             ;clear needed variables and registers
 mov j, 0
 mov di, 0
matr_loop:                            ;loop through matrix elements
 mov dx, [arr+di]
 cmp dx, tmp
 je print_coordinates
raise_ij:                            ;changing variables so that we check next matrix
element
 add di, 2
 inc j
 mov cx, j
 cmp cx, m                           ;if end of row, go to next row
 jl matr_loop
 mov j, 0
 inc i
 mov bx, i
 mov ax, 32
 mul bl
 mov di, ax
```

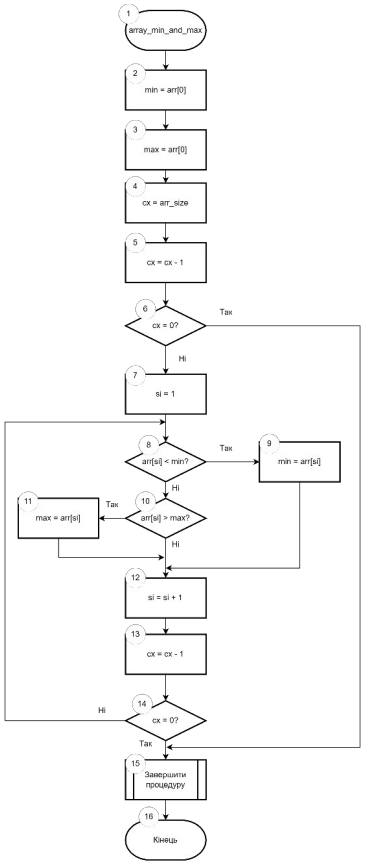
```
 cmp bx, n                      ;if end of last row, end procedure
 jl matr_loop
 ret
print_coordinates:                  ;print i and j indexes of value which entries are
being found
 mov is_in_matrix, 1
 lea dx, matr_tip
 mov ah, 9
 int 21h
 mov ax, i
 inc ax
 mov num, ax
 call result_print
 lea dx, matr_i_tip
 mov ah, 9
 int 21h
 mov ax, j
 inc ax
 mov num, ax
 call result_print
 mov al, ']'
 int 29h
 mov al,10                      ;print a new line after the input
 int 29h
 mov al,13
 int 29h
 jmp raise_ij
find_and_print_entries endp

CSEG ENDS
end main
```

Схема роботи програми 1:


1 Початок
2 Виділення пам'яті під стек
3 Ініціалізація змінних
4 Отримання адреси повернення та ініціалізації DS
5 input_digit
Введення рядка символів
6 str_to_int
Перетворення рядка символів в число
7 is_error = 1? Так A Ні
8 is_error = 2|| num <= 0|| num > 32? Так B Ні
9 Запис числа в arr_size
10 array_input
Введення елементів масиву
11 is_error = 1? Так A Ні
12 is_error = 2? Так B Ні
13 array_sum
Сума елементів масиву
14 result_print_dword
Виведення суми
15 array_min_and_max
Максимальний та мінімальний елемент масиву
16 Записати min в num, вивести підказку
17 result_print
Вивести мінімальний елемент
18 Записати max в num, вивести підказку
19 result_print
Вивести максимальний елемент
20 array_sort
Сортування масиву за зростанням
21 array_print
Виведення масиву
C
A
22 Виведення помилки про некоректні символи
C
B
23 Виведення помилки про невірний діапазон
C
24 Завершити програму
25 Кінець
1 input_digit
2 Виведення підказки
3 Ввід рядка символів
4 Перехід на новий рядок
5 Завершити процедуру
6 Кінець
1 str_to_int
2 Завантаження адреси першого символу; обнулення лічильника циклу та допоміжних змінних
D
3 Кінець рядку? Так Ні
4 Символ менше "0"? Так Ні
5 Символ більше "9"? Так F Ні
6 Збільшити лічильник циклу; перейти до наступного символу
E
7 Символ є "-"? Ні F Так
8 Цей символ перший? Ні F Так
9 is_negative = 1
10 Збільшити лічильник циклу; перейти до наступного символу
D
E
11 Перетворити символ в цифру
12 num = num * 10
13 CF = 1 || SF = 1? Так G Ні
14 num = 32760? Так Ні
15 num = num + {цифра}
16 CF = 1 || SF = 1? Ні D Так
G
17 is_negative = 0? Так Ні
18 num = -num
19 num = num - ax
20 OF = 1? Так G
H
F
21 is_error = 1
H
G
22 is_error = 2
H
23 is_negative = 1? Так 24 num = -num Ні
H
25 Завершити процедуру
26 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
1 result_print_dword
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







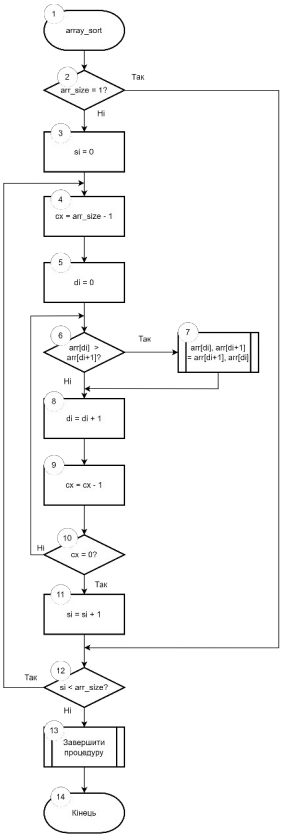
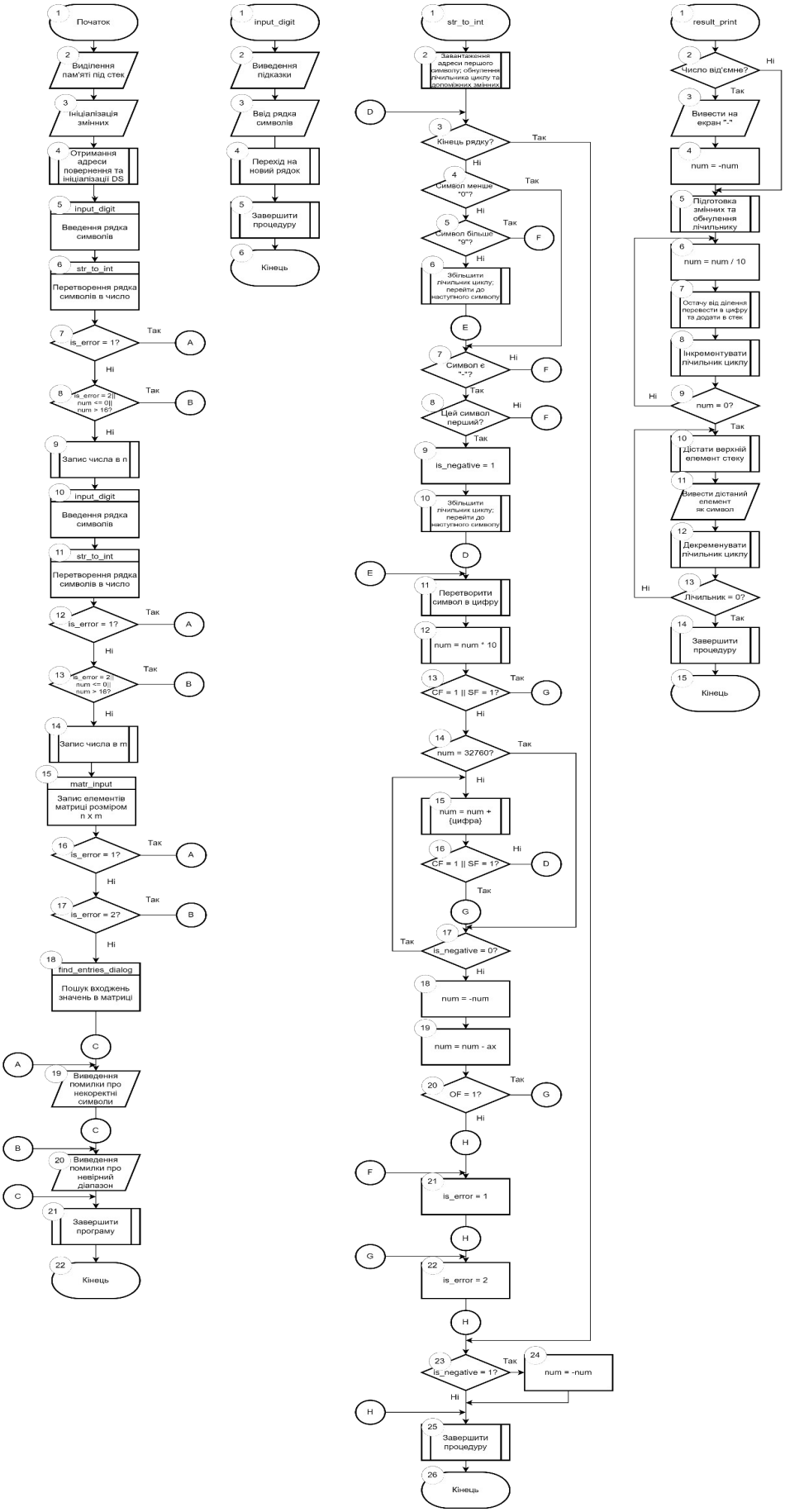
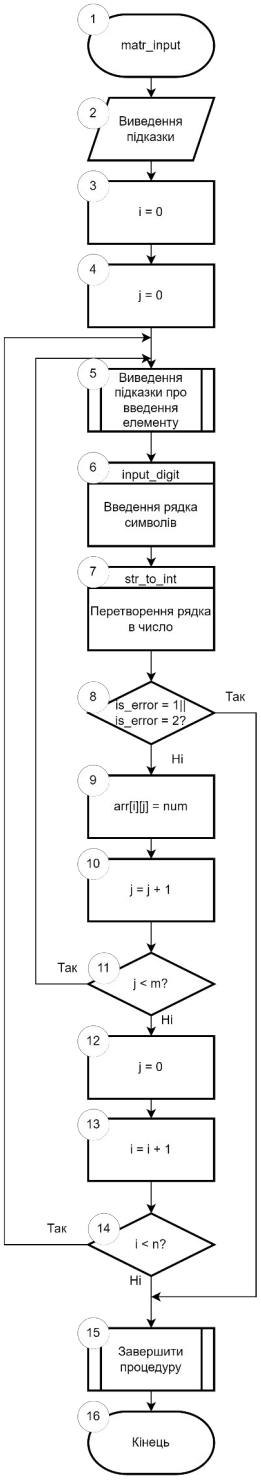
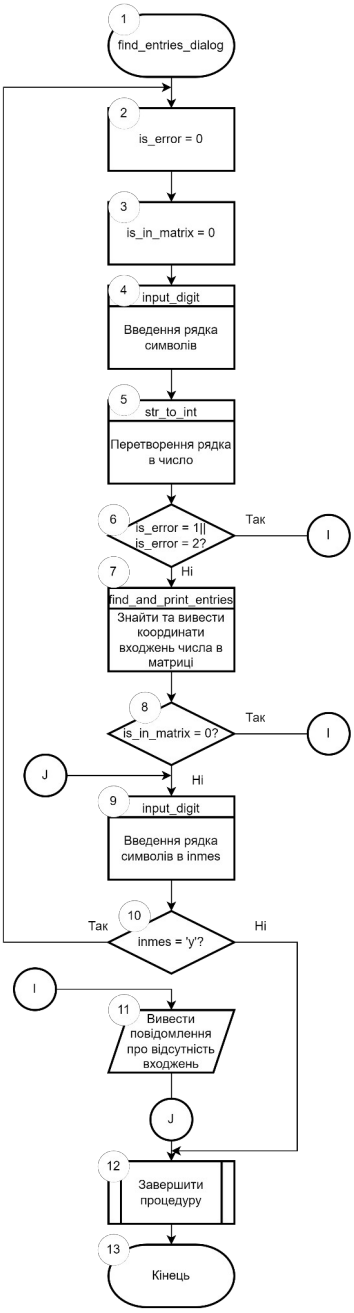
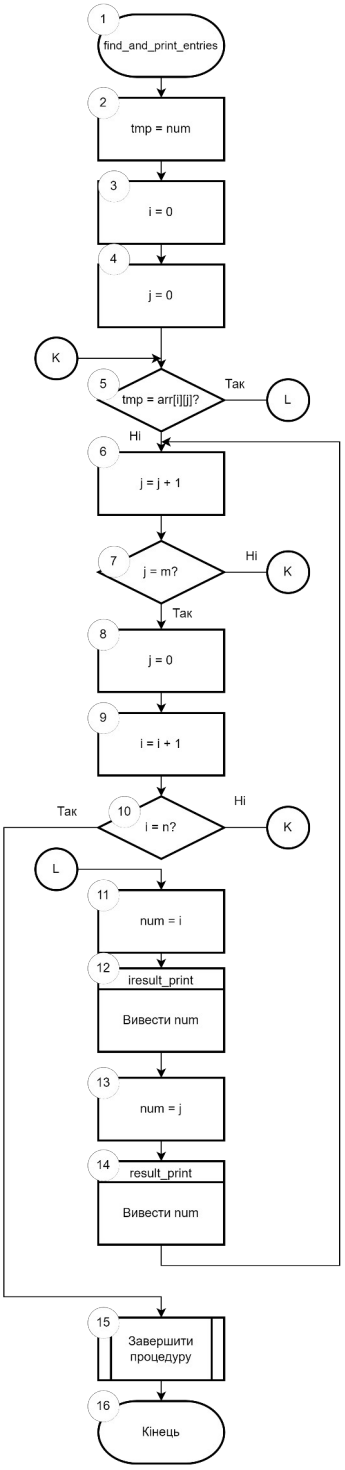


Схема роботи програми2:


1 Початок
2 Виділення пам'яті під стек
3 Ініціалізація змінних
4 Отримання адреси повернення та ініціалізації DS
5 input_digit
Введення рядка символів
6 str_to_int
Перетворення рядка символів в число
7 is_error = 1?
Так
A
Ні
8 is_error = 2|| num <= 0|| num > 16?
Так
B
Ні
9 Запис числа в n
10 input_digit
Введення рядка символів
11 str_to_int
Перетворення рядка символів в число
12 is_error = 1?
Так
A
Ні
13 is_error = 2|| num <= 0|| num > 16?
Так
B
Ні
14 Запис числа в m
15 matr_input
Запис елементів матриці розміром n X m
16 is_error = 1?
Так
A
Ні
17 is_error = 2?
Так
B
Ні
18 find_entries_dialog
Пошук входжень значень в матриці
C
A
19 Виведення помилки про некоректні символи
C
B
20 Виведення помилки про невірний діапазон
C
21 Завершити програму
22 Кінець
1 input_digit
2 Виведення підказки
3 Ввід рядка символів
4 Перехід на новий рядок
5 Завершити процедуру
6 Кінець
1 str_to_int
2 Завантаження адреси першого символу; обнулення лічильника циклу та допоміжних змінних
D
3 Кінець рядку?
Так
Ні
4 Символ менше "0"?
Так
Ні
5 Символ більше "9"?
Так
F
Ні
6 Збільшити лічильник циклу; перейти до наступного символу
E
7 Символ є "-"?
Ні
F
Так
8 Цей символ перший?
Ні
F
Так
9 is_negative = 1
10 Збільшити лічильник циклу; перейти до наступного символу
D
E
11 Перетворити символ в цифру
12 num = num * 10
13 CF = 1 || SF = 1?
Так
G
Ні
14 num = 32760?
Так
Ні
15 num = num + {цифра}
16 CF = 1 || SF = 1?
Ні
D
Так
G
17 is_negative = 0?
Так
Ні
18 num = -num
19 num = num - ax
20 OF = 1?
Так
G
Ні
H
F
21 is_error = 1
H
G
22 is_error = 2
H
23 is_negative = 1?
Так
24 num = -num
Ні
H
25 Завершити процедуру
26 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 Декременувати лічильник циклу
13 Лічильник = 0?
Ні
Так
14 Завершити процедуру
15 Кінець

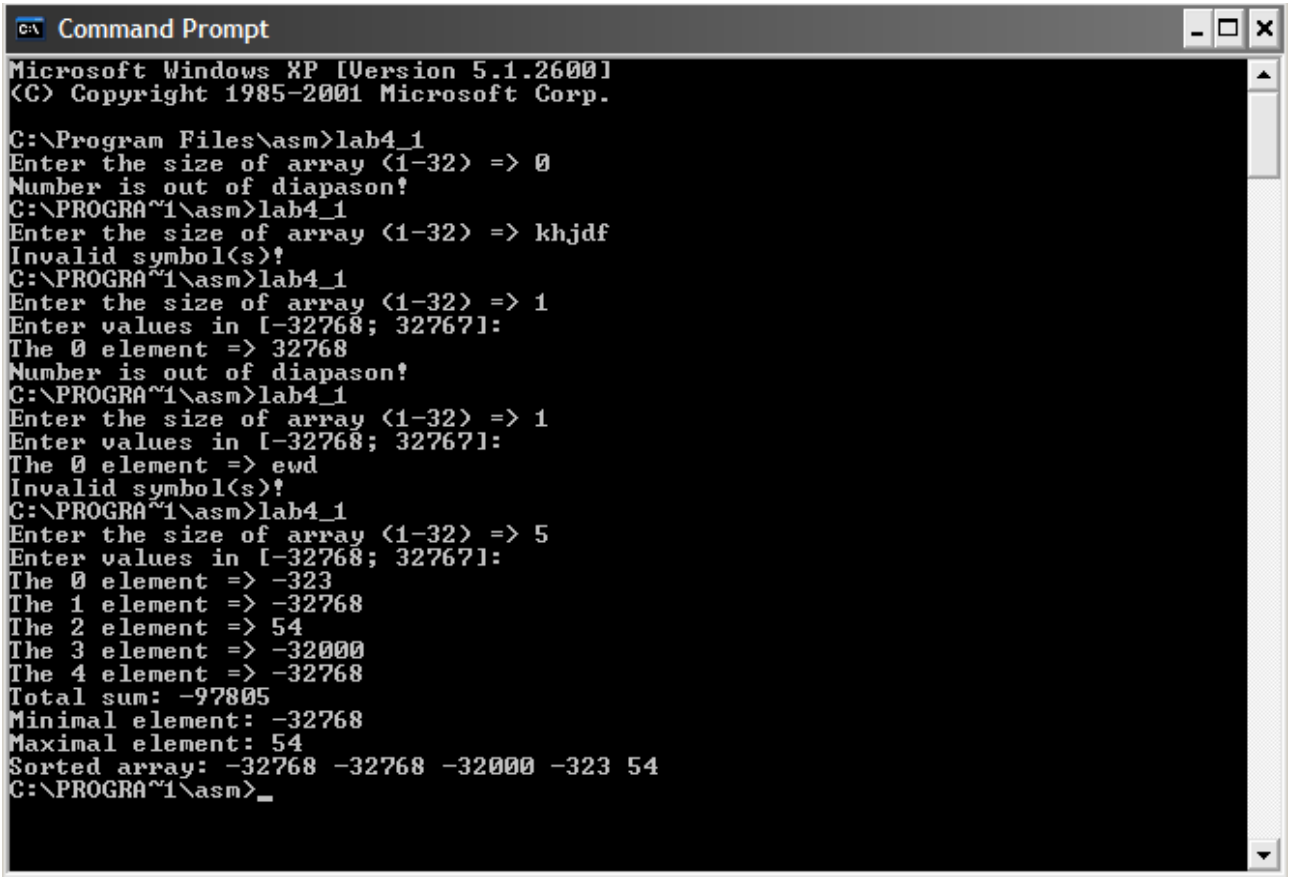






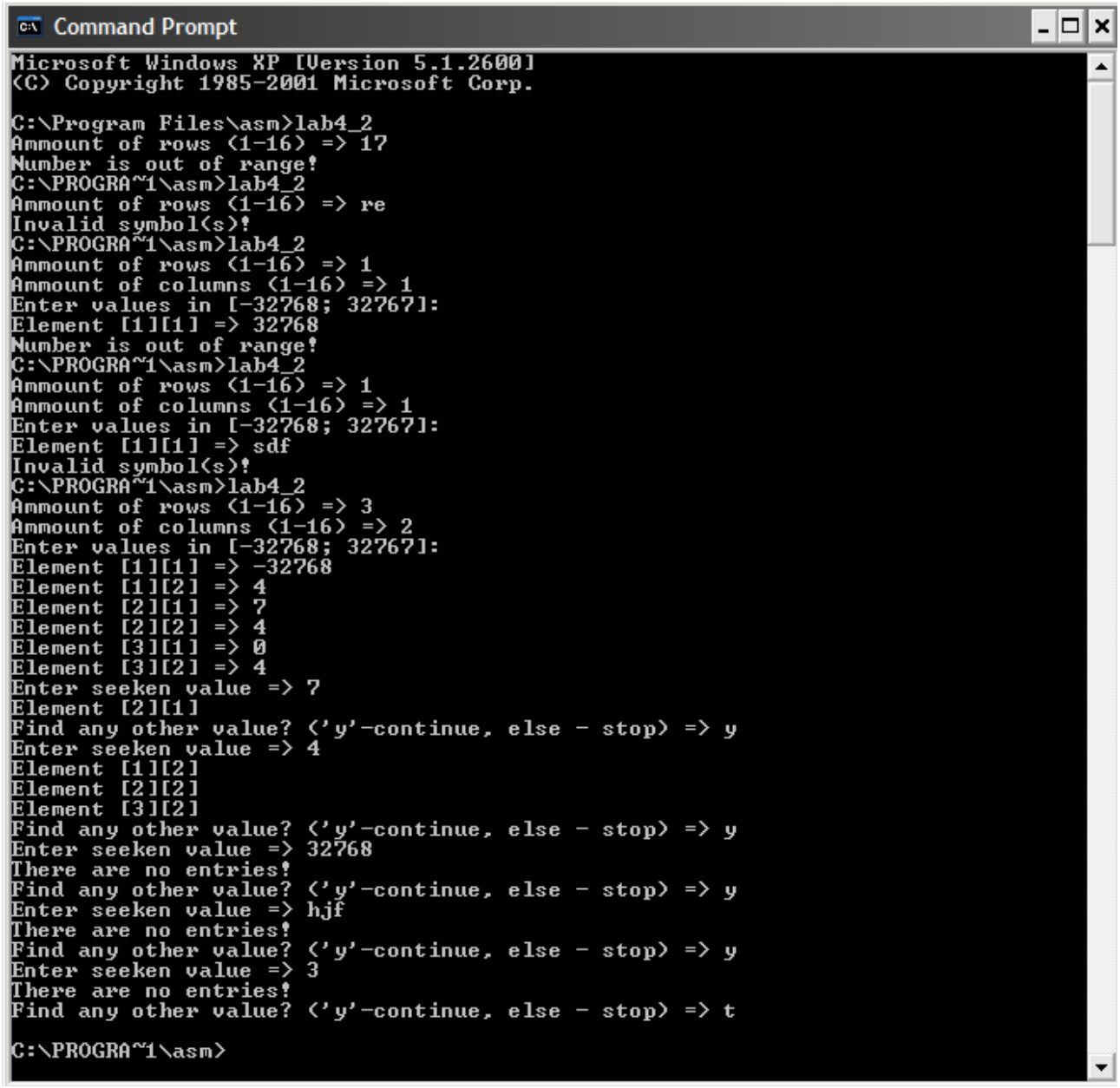
Тестування роботи програм:

Програма 1:



Перші чотири рази програма вивела помилки «Число поза дозволеного діапазону» чи «Використано невірні символи». П’ятого 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.

Програма 2:



Перші чотири рази програма вивела помилки «Число поза дозволеного діапазону» чи «Використано невірні символи». П’ятого разу було продемонстровано, як програма працює при умові введення правильних значень. Спочатку було введено користувачем розмір та значення матриці, а далі програма запитала, координати якого значення шукати. Якщо такого елемента нема (не є в переліку елементів матриці, чи число не відповідає можливому діапазону значень чи введено взагалі не число), то програма повідомляє, що це значення не зустрічається в матриці. Якщо таке значення зустрічається, то програма виводить координати всіх елементів матриці, що дорівнюють вказаному числу. Далі програма пропонує продовжити пошук елементів у матриці, користувач може продовжити чи завершити виконання програми.

Висновок:

1. Написана програма, яка має наступний функціонал:

1) Можливість введення користувачем розміру одномірного масиву [1-32].

2) Можливість введення користувачем значень елементів одномірного масиву у діапазоні [-32768; 32767].
3) Можливість знаходження суми елементів одномірного масиву (навіть якщо сума виходить за межі [-32768; 32767].
4) Можливість пошуку максимального та мінімального елемента одномірного масиву.
5) Можливість сортування одномірного масиву цілих чисел загального вигляду за зростанням.
Код та приклад роботи наведені у звіті.
2. Написана програма, яка має наступний функціонал:
1) Можливість введення користувачем розміру двомірного масиву [1-16][1-16].
2) Можливість введення користувачем значень елементів двомірного масиву в діапазоні [-32768; 32767].
3) Можливість пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий елемент вводить користувач, навіть якщо входжень декілька.
3. Програма має захист від некоректного введення вхідних даних (символи та переповнення).