

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з комп'ютерного практикуму № 3 з дисципліни

«Системне програмне забезпечення»

«Програмування розгалужених алгоритмів»

Виконав:
студент групи ІП-11
Лесів В. І.

Перевірив:
Лісовиченко О. І.
«12» травня 2023 р.

Київ 2023

Комп'ютерний практикум 3

Програмування розгалужених алгоритмів

Постановка завдання.

Написати програму, яка повинна мати наступний функціонал:

1. Можливість введення користувачем значень x , y , t , a , b за необхідності.
2. Обчислювати значення функції за введеними значеннями.
3. Виводити на екран результат обчислень.
4. Якщо є ділення, то результат дозволяється виводити:
 - a. як дійсне число – підвищена складність;
 - b. окремо цілу частину та остачу (наприклад: 1 остача 2) – середня складність;
 - c. окремо цілу частину та остачу як дріб – середня складність.
5. Програма повинна мати захист від некоректного введення вхідних даних (символи, переповнення, ділення на 0 і т.і.)

Варіант №14.

$$Z = \begin{cases} \frac{35}{x} + x^3 & (1 < x \leq 3) \\ \frac{x}{1+x^2} & (1 < x \leq 1) \\ 2x & (x \leq -1) \\ x + y & (\text{в інших випадках}) \end{cases}$$

Хід роботи.

Текст програми.

```
STSEG SEGMENT PARA STACK "STACK"
```

```
DB 64 DUP("STACK")
```

```
STSEG ENDS
```

```
DSEG SEGMENT PARA PUBLIC "DATA"
```

```
x dw 0
```

```
y dw 0
```

```
input_number db 7,?,7 dup (" $")
```

```
welcome db "Enter numbers: $"
```

```
enter_x db 13, 10, "x = $"
```

```
enter_y db "y = $"
```

```
is_negative db 0
```

```
number dw 0
```

```
digit dw 0
```

```
is_error db 0
```

```
error_msg db "Error!$"
```

```
remainder dw 0
```

```
remainder_msg db " remainder $"
```

```
DSEG ENDS
```

```
CSEG SEGMENT PARA PUBLIC "CODE"
```

```
ASSUME CS:CSEG, DS:DSEG, SS:STSEG
```

main proc

mov ax, dseg

mov ds, ax

LEA dx, welcome

MOV ah,9

INT 21h

; ВВОДИМО X

lea dx, enter_x

call read

call transform

cmp is_error, 1

je error

; записуємо значення number до x і обнуляємо number

mov ax, number

mov x, ax

mov number, 0

mov is_negative, 0

; якщо $x \leq 3$, то y не потрібен

cmp x, 3

jle without_y

; вВОДИМО у

lea dx, enter_y

call read

call transform

cmp is_error, 1

je error

; записуємо number в у і обнуляємо

mov ax, number

mov y, ax

mov number, 0

without_y:

call calculate

cmp is_error, 1

je error

call output_result

cmp remainder, 0

je end_program

LEA dx, remainder_msg

MOV ah,9

INT 21h

```
    mov ax, remainder

    mov number, ax

    call output_result

    jmp end_program

error:

    LEA dx, error_msg

    MOV ah,9

    INT 21h

end_program:

    mov AH, 4CH

    int 21H

    ret

main endp
```

```
read proc
```

```
; показ повідомлення
```

```
    mov ah, 9
```

```
    int 21h
```

```
    lea dx, input_number
```

```
    mov ah, 10
```

```
    int 21h
```

; переходимо на наступний рядок після вводу

```
mov al,10
```

```
int 29h
```

```
mov al,13
```

```
int 29h
```

```
ret
```

```
read endp
```

```
transform proc
```

```
mov si, offset input_number + 2 ; завантажуюмо адресу input_number +
2, тобто перший елемент
```

```
mov cx, 0
```

```
convert_loop:
```

```
; перевіряємо, чи кінець рядка
```

```
mov ax, 0
```

```
mov al, input_number + 1
```

```
cmp al, cl
```

```
je finish
```

```
mov al, [si] ; якщо ні, беремо символ цього рядка
```

```
cmp al, '0'
```

```
jnl negative_sign
```

```
cmp al, '9'
```

jg error1

inc cx

inc si ; переходимо до наступного елемента

jmp transform_number

negative_sign:

cmp al, '-'

jne error1

; перевіряємо, чи - стоїть на початку рядка

cmp cx, 0

jne error1

mov is_negative, 1

inc cx

inc si

jmp convert_loop

transform_number:

sub al, '0'

mov digit, ax

mov bx, 10

mov ax, number

mul bx

jc error1


```
js error1  
  
mov number, ax  
  
mov ax, digit  
  
add number, ax  
  
jc error1  
  
js error1  
  
jmp convert_loop
```

```
error1:  
  
mov is_error, 1  
  
jmp end_prog
```

```
finish:  
  
cmp is_negative, 1  
  
jne end_prog  
  
neg number
```

```
end_prog:  
  
ret
```

```
transform endp
```

```
calculate proc
```

```
cmp x, -1
```

jle case3

cmp x, 1

jle case2

cmp x, 3

jle case1

jmp case4

case1:

; $z = 35/x + x^3 = 35/x + x * x * x$

mov ax, 35

div x

mov number, ax

mov remainder, dx

mov bx, x

mov ax, x

mul bx

jc error2

mov bx, x

mul bx

jc error2

add ax, number

mov number, ax

jmp calculated

case2:

; $z=x/(1+x*x)$

mov bx, x

mov ax, x

imul bx

jo error2

add ax, 1

mov number, ax

mov ax, x

idiv number

mov remainder, dx

mov number, ax

jmp calculated

case3:

; $z=2*x$

mov bx, 2

mov ax, x

imul bx

jo error2

mov number, ax

jmp calculated

case4:

; z=x+y

mov ax,x

add ax,y

jo error2

mov number, ax

jmp calculated

error2:

mov is_error, 1

calculated:

ret

calculate endp

output_result proc

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

ret

output_result endp

CSEG ENDS

end main
```

Схема функціонування програми.

Рисунок 1. Схема основної частини програми.

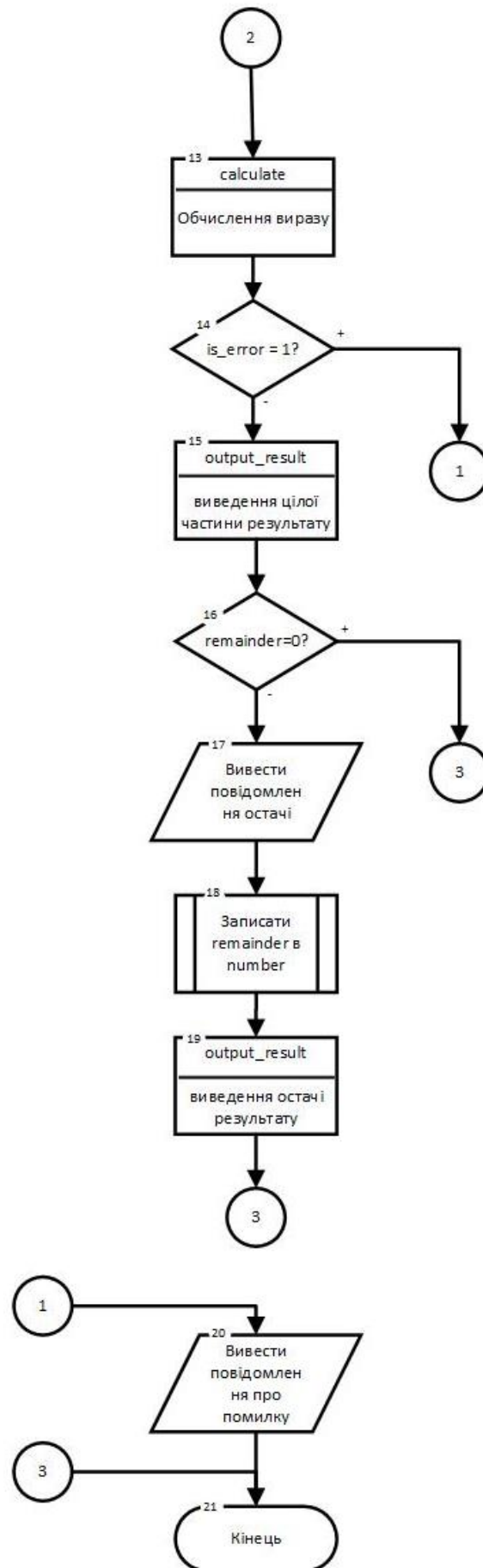


Рисунок 2. Схема основної частини програми (продовження).



Рисунок 2. Схема процедури read

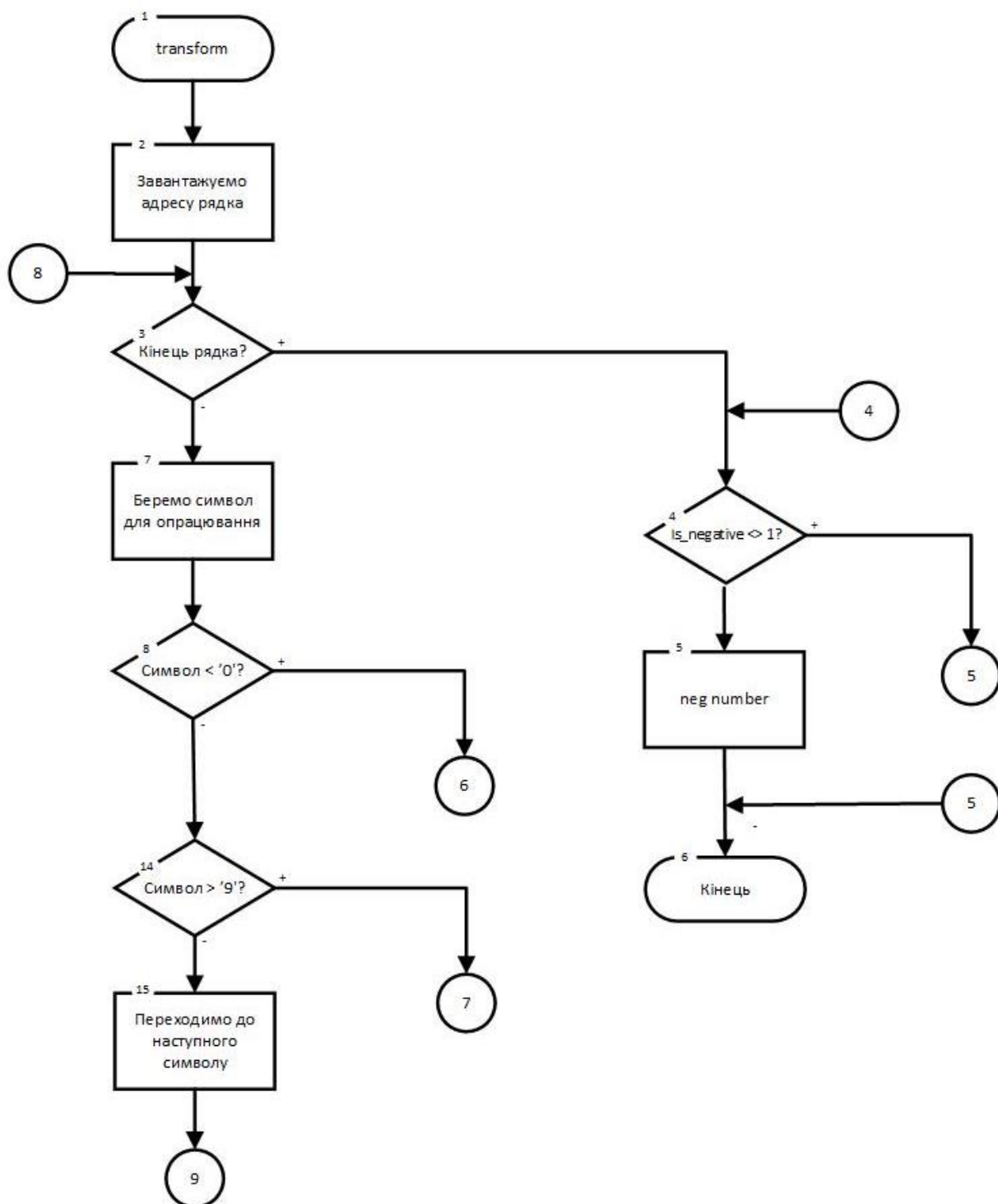


Рисунок 4. Схема процедури transform

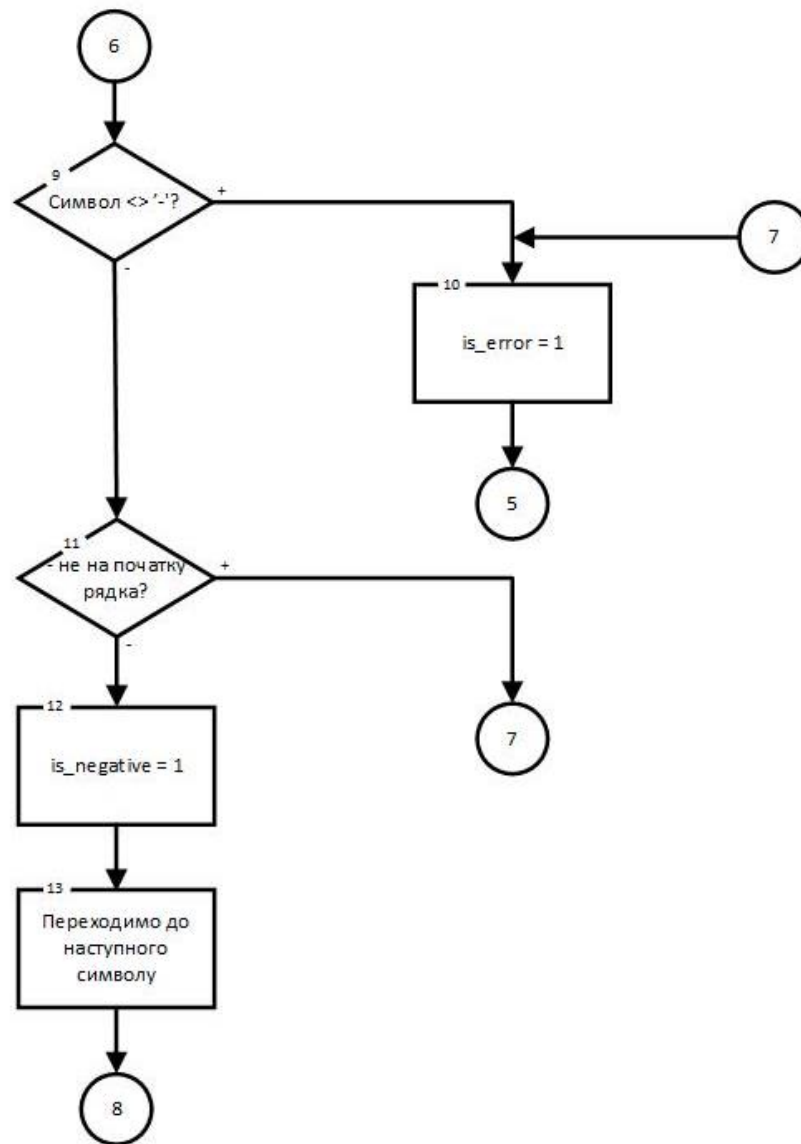


Рисунок 5. Схема процедури transform (продовження 1)

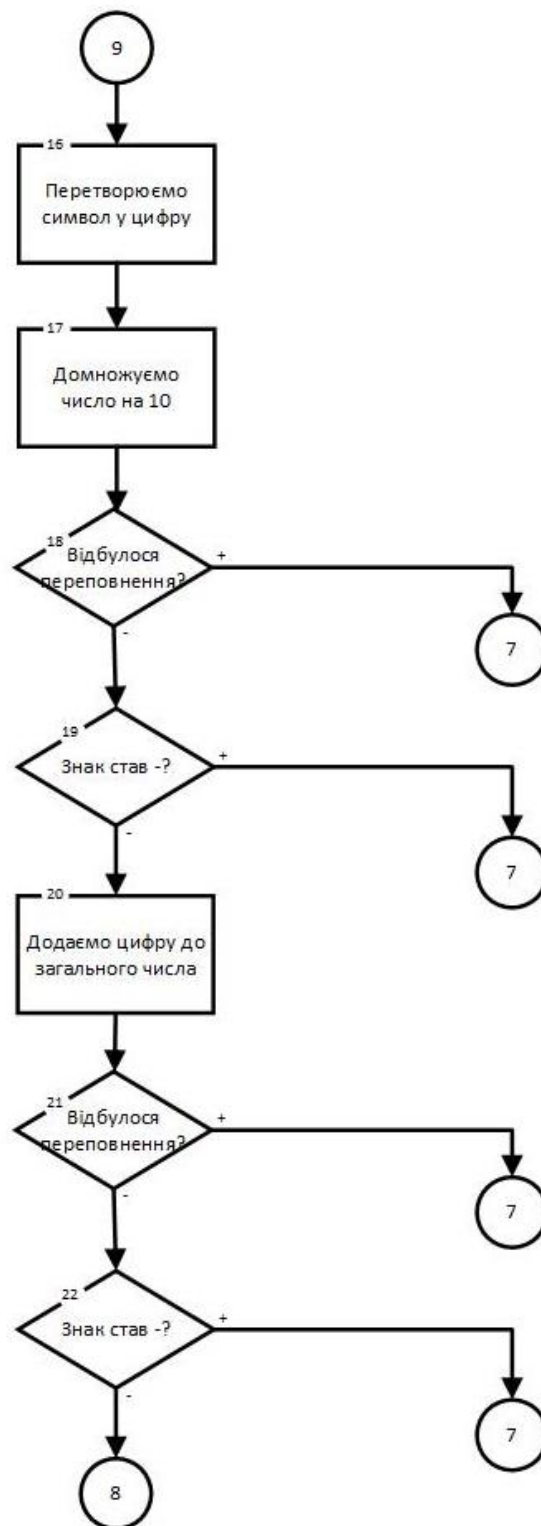


Рисунок 6. Схема процедури transform (продовження 2)

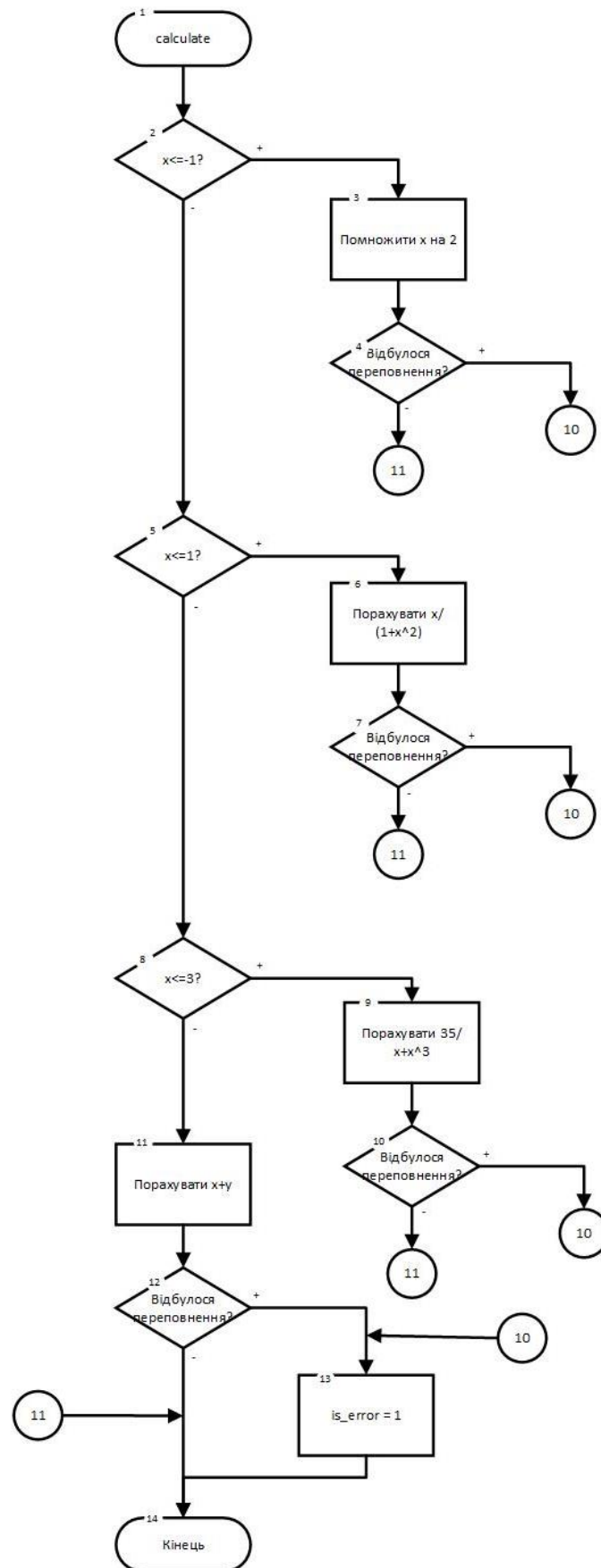
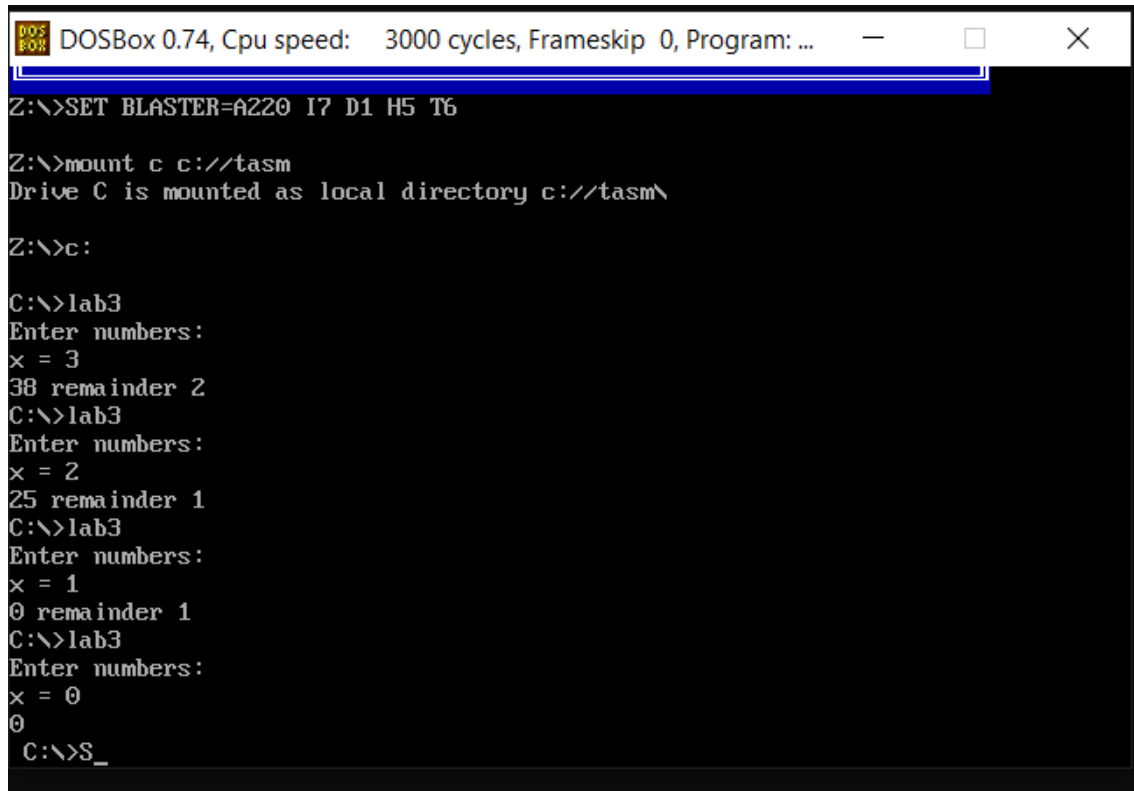


Рисунок 4. Схема процедури calculate



Рисунок 5. Схема процедури output_result

Приклад виконання програми.



```

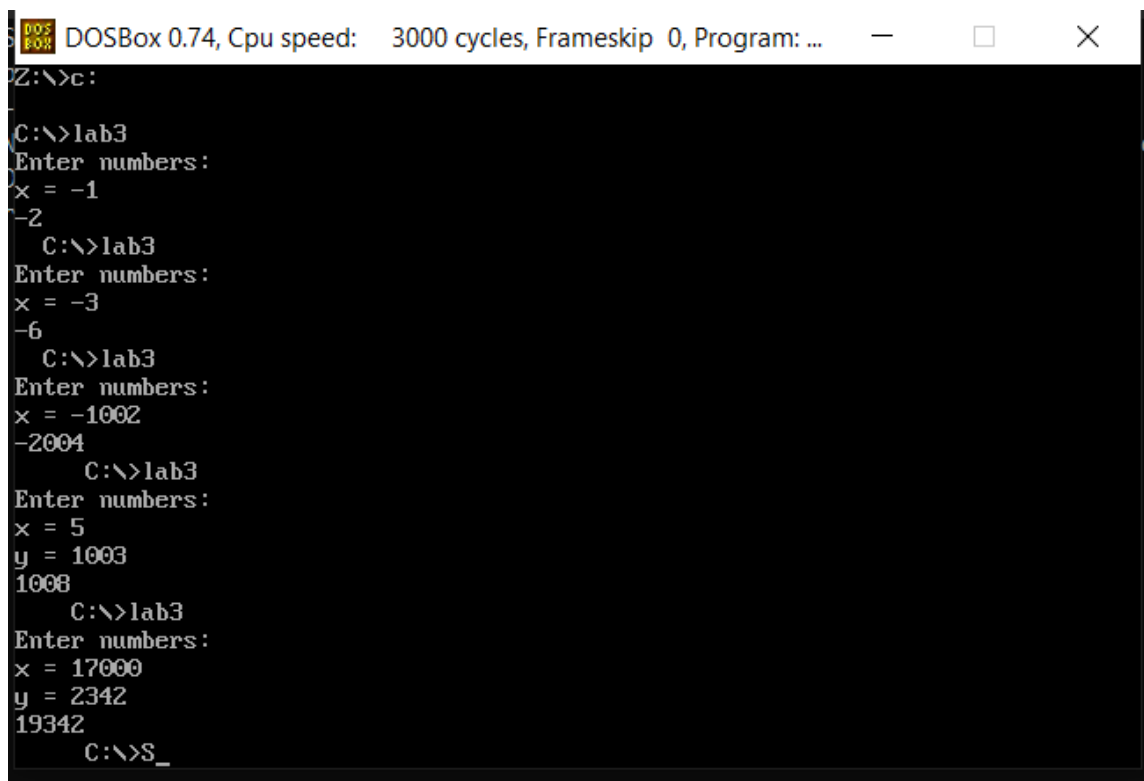
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c c://tasm
Drive C is mounted as local directory c://tasm\

Z:\>c:

C:\>lab3
Enter numbers:
x = 3
38 remainder 2
C:\>lab3
Enter numbers:
x = 2
25 remainder 1
C:\>lab3
Enter numbers:
x = 1
0 remainder 1
C:\>lab3
Enter numbers:
x = 0
0
C:\>S_
  
```

Рисунок 6. Приклад виконання програми з $1 < x \leq 3$ і з $-1 < x \leq 1$.

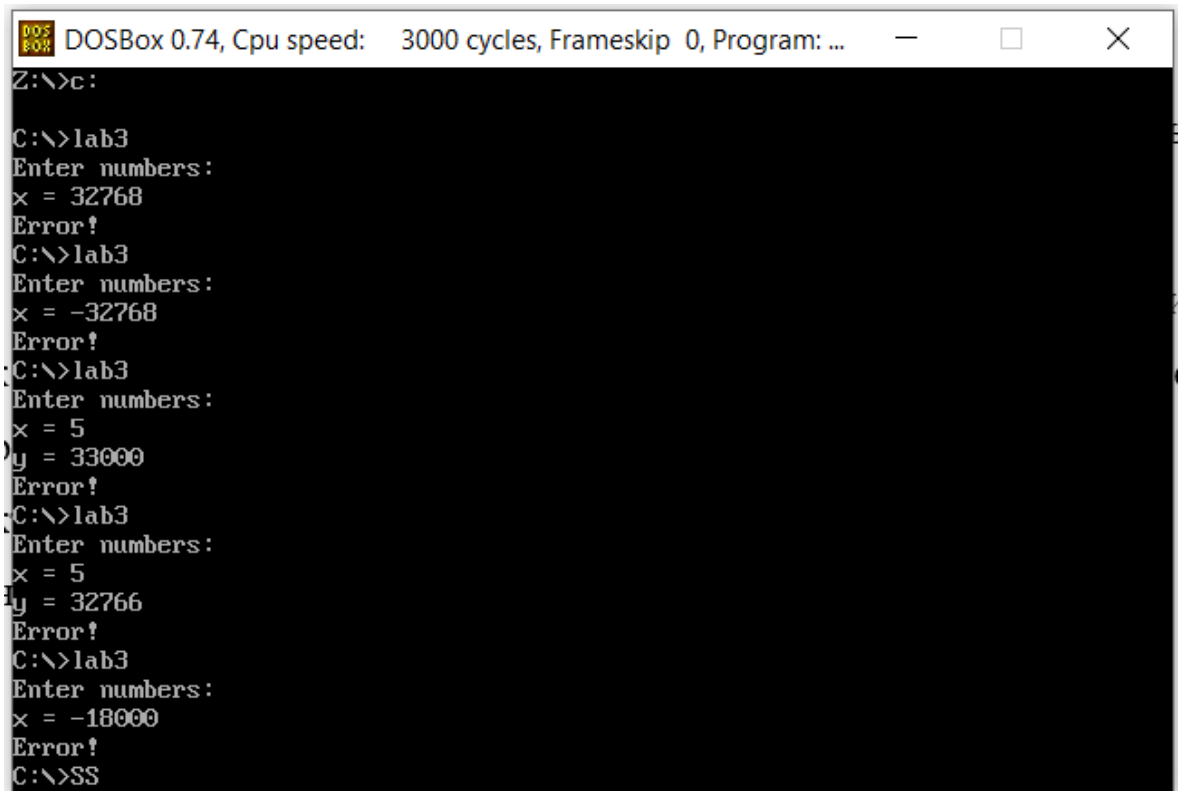


```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Z:\>c:

C:\>lab3
Enter numbers:
x = -1
-2
C:\>lab3
Enter numbers:
x = -3
-6
C:\>lab3
Enter numbers:
x = -1002
-2004
C:\>lab3
Enter numbers:
x = 5
y = 1003
1008
C:\>lab3
Enter numbers:
x = 17000
y = 2342
19342
C:\>S_
  
```

Рисунок 7. Приклад виконання програми з $x \leq -1$ і з $x > 3$

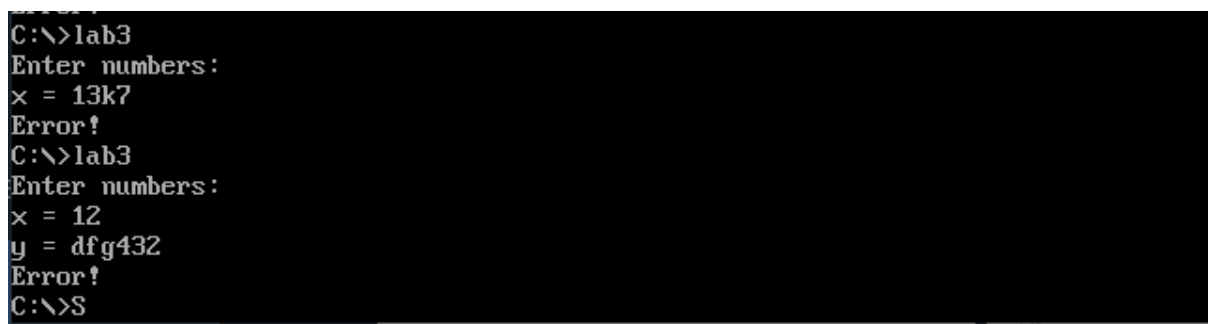


```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
Z:\>c:
C:\>lab3
Enter numbers:
x = 32768
Error!
C:\>lab3
Enter numbers:
x = -32768
Error!
C:\>lab3
Enter numbers:
x = 5
y = 33000
Error!
C:\>lab3
Enter numbers:
x = 5
y = 32766
Error!
C:\>lab3
Enter numbers:
x = -18000
Error!
C:\>SS

```

Рисунок 8. Приклад виконання програми, коли числа виходять за межі допустимих



```

C:\>lab3
Enter numbers:
x = 13k7
Error!
C:\>lab3
Enter numbers:
x = 12
y = dfg432
Error!
C:\>S

```

Рисунок 9. Приклад виконання програми, коли вводяться не числа

Програма повідомляє про некоректне введення, якщо введено символи або ж значення, що перевищує задані межі. Якщо значення некоректне, програма виводить повідомлення й завершує роботу. Після введення коректного значення програма виводить повідомлення про результат виконання.

Перевірні дані – розрахунки.

<i>Вхідні дані</i>	<i>Розрахунки</i>	<i>Вихідні дані</i>
x = 3	$f(3) = \frac{35}{3} + 3^3 = 38 \text{ (ост. 2)}$	38 remainder 2
x = 2	$f(2) = \frac{35}{2} + 2^3 = 25 \text{ (ост. 1)}$	25 remainder 1
x = 1	$f(1) = \frac{1}{1 + 1^2} = 0 \text{ (ост. 1)}$	0 remainder 1
x = 0	$f(0) = \frac{0}{1 + 0^2} = 0$	0
x = -1	$f(-1) = 2 * (-1) = -2$	-2
x = -3	$f(-3) = 2 * (-3) = -6$	-6
x = -1002	$f(-1002) = 2 * (-1002) = -2004$	-2004
x = 5 y = 1003	$f(5, 1003) = 5 + 1003 = 1008$	1008
x = 17000 y = 2342	$f(17000, 2342) = 17000 + 2342 = 19342$	19342
x = 32768	Помилка переповнення	Error!
x = -32768	Помилка переповнення	Error!
x = 5 y = 33000	Помилка переповнення для y	Error!
x = 5 y = 32766	Помилка переповнення для результату обчислень	Error!
x = -18000	Помилка переповнення для результату обчислень	Error!
x = 13k7	Помилка: введено символи, окрім цифр	Error!
x = 12 y = dfg432	Помилка: введено символи, окрім цифр, в y	Error!

Висновок.

Отже, у даній роботі я ознайомився із програмуванням розгалужених алгоритмів.

Було написано програму, яка має наступний функціонал:

1. Можливість введення користувачем значень x та y за необхідності.
2. Обчислює значення функції за введеними значеннями відповідно до варіанту.
3. Виводить на екран результат обчислень.
 - а. Якщо є ділення, то результат виводиться окремо цілу частину та остачу.

Програма має захист від некоректного введення вхідних даних (символи, переповнення). Програму було скомпільовано, налагоджено та виконано. У результаті виконання програми отримується коректний й очікуваний результат, відповідно до попередніх розрахунків.