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.**

**“20” травня 2023 р.**

Київ – 2023

**Комп’ютерний практикум №3**

**Тема**: програмування розгалужених алгоритмів

**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:

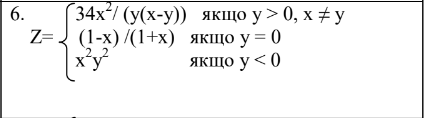
а) як дійсне число (наприклад: 5 3 = 1,666667) – підвищена складність;

б) окремо цілу частину та остачу (наприклад: 5 3 = 1 остача 2) – середня складність;

в) окремо цілу частину та остачу як дріб (наприклад: 5 3 = 1 2 3 ) – середня складність.

5. Програма повинна мати захист від некоректного введення вхідних даних (символи, переповнення, ділення на 0 і т.і.).

Варіант 6:



Код програми:  
sseg segment para stack 'stack'

db 64 dup ( 'STACK' )

sseg ends

dseg segment para public 'data'

num dw 0

n dw 0

den dw 0

temp dw 0

remainder dw 0

var\_x dw 0

var\_y dw 0

result dw 0

is\_negative dw 0

; prompt db 0dh, 0ah, 'enter number in range of -32734 to 32767 : $'

enterX db 0dh, 0ah, 'enter x in (-32,768; 32767] : $'

enterY db 0dh, 0ah, 'enter y in (-32,768; 32767]: $'

printresult db 0dh, 0ah, 'result: $'

printX db 0dh, 0ah, 'X is $'

printY db 0dh, 0ah, 'Y is $'

error\_msg db 0dh, 0ah, 'Error $'

error\_msg1 db 0dh, 0ah, 'the divided is greater than the divisor $'

overflow\_error db 'Overflow $'

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

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

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

equals\_0 db 0dh, 0ah, '(1-x)/(1+x) = $'

less\_than\_0 db 0dh, 0ah, 'x^2 \* y^2 = $'

greater\_than\_0 db 0dh, 0ah, '34x^2 / (y(x-y))= $'

div\_remainder db ' remainder: $'

dseg ends

cseg segment para public 'code'

main proc

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

push ds

xor ax, ax

push ax

; initializing DS

mov ax, dseg

mov ds, ax

lea dx, enterX

mov ah, 09h

int 21h

call read\_digit

mov ax,num

mov var\_x, ax

call print\_var\_x

lea dx, enterY

mov ah, 09h

int 21h

call read\_digit

mov ax,num

mov var\_y,ax

call print\_var\_y

cmp var\_y, 0

jg func\_greater\_0

je func\_equals\_0

jmp func\_less\_0

func\_greater\_0:

lea dx,greater\_than\_0

mov ah,09h

int 21h

call function\_greater\_than\_0

mov ah, 4ch

int 21h

func\_less\_0:

lea dx, less\_than\_0

mov ah, 09h

int 21h

call function\_less\_than\_0

mov ah, 4ch

int 21h

func\_equals\_0:

lea dx, equals\_0

mov ah, 09h

int 21h

call function\_equals\_0

mov ah, 4ch

int 21h

main endp

function\_greater\_than\_0 proc

xor ax,ax

xor cx,cx

mov ax, var\_x

CMP ax,var\_y

jz error1

mov ax, var\_x

mov cx, var\_x

IMUL cx ;AX= x^2

jc func3\_overflow

jo func3\_overflow

mov result,ax; res = x^2

xor ax,ax

MOV AX,34

IMUL result ; AX=34x^2

jc func3\_overflow

jo func3\_overflow

mov result,ax; res = 34x^2

xor ax,ax

MOV BX,var\_x

SUB BX,var\_y ; x-y

MOV AX,BX

MOV BX,var\_y

IMUL BX;AX= y(x-y)

jc func3\_overflow

jo func3\_overflow

mov cx,ax;cx= y(x-y)

mov ax,result

cwd

idiv cx ;ax=34x^2/y(x-y) ////dx

jc func3\_overflow

jo func3\_overflow

mov result, ax

mov remainder, dx

call print\_result

cmp remainder,0

jne func3\_remainder

ret

func3\_remainder:

lea dx, div\_remainder

mov ah, 09h

int 21h

call print\_num\_remainder

ret

func3\_overflow:

lea dx, overflow\_error

mov ah, 09h

int 21h

ret

error1:

lea dx, error\_msg

mov ah, 09h

int 21h

ret

function\_greater\_than\_0 endp

function\_equals\_0 proc

cmp var\_x, -1

jz error

; calc

xor ax,ax

mov ax,1

sub ax,var\_x;1-x

mov n,ax

mov ax,1

add ax,var\_x ; 1+x

mov den,ax

mov ax,n

mov bx,den

cwd

idiv bx

MOV result, AX

mov remainder, dx

;jc error\_1

;jo error\_1

call print\_result

cmp remainder,0

jne func2\_remainder

ret

func2\_remainder:

lea dx, div\_remainder

mov ah, 09h

int 21h

call print\_num\_remainder

ret

func2\_overflow:

lea dx, overflow\_error

mov ah, 09h

int 21h

ret

error:

mov ah, 09h

lea dx, error\_msg

int 21h

ret

error\_1:

mov ah, 09h

lea dx, error\_msg1

int 21h

ret

function\_equals\_0 endp

function\_less\_than\_0 proc

xor ax,ax

xor cx,cx

; calc y<0

MOV AX, var\_x

IMUL var\_x

jo func\_overflow

MOV BX, var\_y

IMUL var\_y

jo func\_overflow

IMUL BX

jo func\_overflow

MOV result, AX

call print\_result

ret

func\_overflow:

lea dx, overflow\_error

mov ah, 09h

int 21h

ret

function\_less\_than\_0 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

mul 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:

lea dx, input\_error\_message

mov ah, 09h

int 21h

mov ah, 4ch

int 21h

wrong\_character:

lea dx, wrong\_character\_message

mov ah, 09h

int 21h

mov ah, 4ch

int 21h

empty\_input:

lea dx, empty\_input\_message

mov ah, 09h

int 21h

mov ah, 4ch

int 21h

read\_digit endp

print\_num\_remainder proc

mov bx, remainder

or bx, bx

jns mre1

mov al, '-'

int 29h

neg bx

mre1:

mov ax, bx

xor cx, cx

mov bx, 10

mre2:

xor dx, dx

div bx

add dl, '0'

push dx

inc cx

test ax, ax

jnz mre2

mre3:

pop ax

int 29h

loop mre3

ret

print\_num\_remainder endp

print\_result proc

mov bx, result

or bx, bx

jns mr1

mov al, '-'

int 29h

neg bx

mr1:

mov ax, bx

xor cx, cx

mov bx, 10

mr2:

xor dx, dx

div bx

add dl, '0'

push dx

inc cx

test ax, ax

jnz mr2

mr3:

pop ax

int 29h

loop mr3

ret

print\_result endp

print\_var\_x proc

lea dx, printX

mov ah, 09h

int 21h

mov bx, var\_x

or bx, bx

jns mx1

mov al, '-'

int 29h

neg bx

mx1:

mov ax, bx

xor cx, cx

mov bx, 10

mx2:

xor dx, dx

div bx

add dl, '0'

push dx

inc cx

test ax, ax

jnz mx2

mx3:

pop ax

int 29h

loop mx3

ret

print\_var\_x endp

print\_var\_y proc

lea dx, printY

mov ah, 09h

int 21h

mov bx, var\_y

or bx, bx

jns my1

mov al, '-'

int 29h

neg bx

my1:

mov ax, bx

xor cx, cx

mov bx, 10

my2:

xor dx, dx

div bx

add dl, '0'

push dx

inc cx

test ax, ax

jnz my2

my3:

pop ax

int 29h

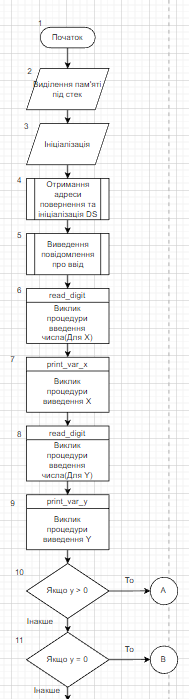
loop my3

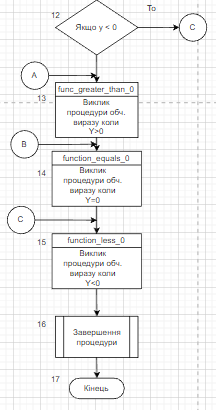
ret

print\_var\_y endp

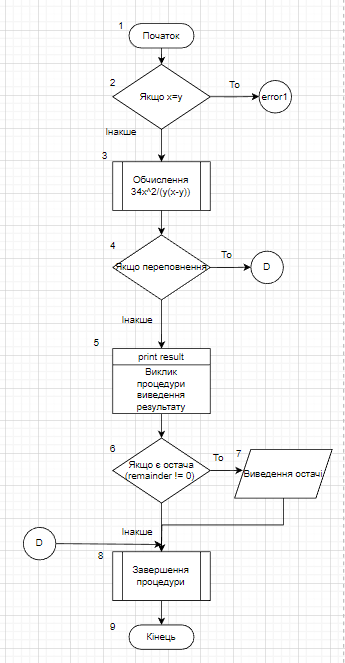
cseg ends

end main

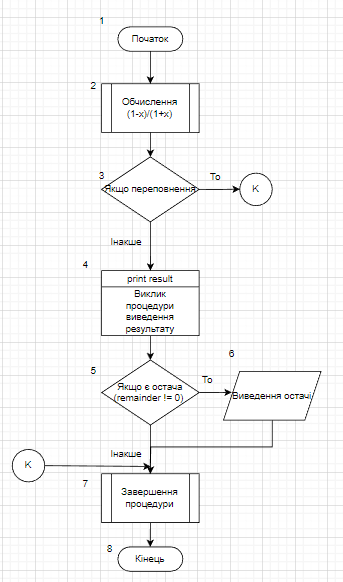
Блок-схема програми:  
 



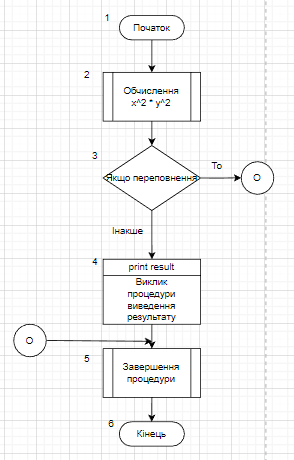
Блок-схема процедури function\_greater\_0:



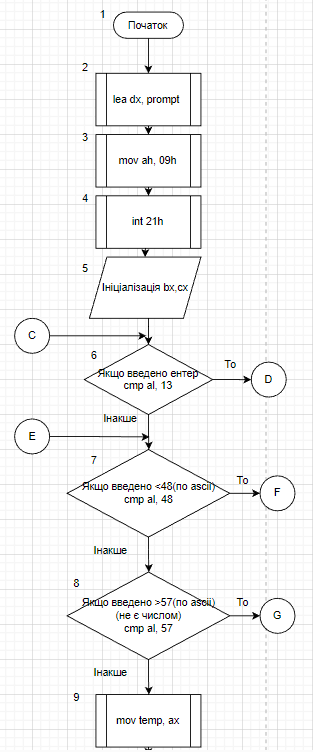
Блок-схема процедури function\_equals\_0:

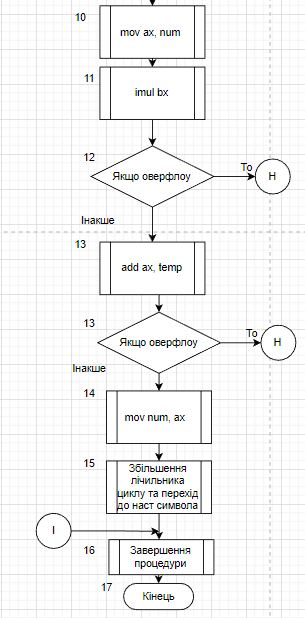


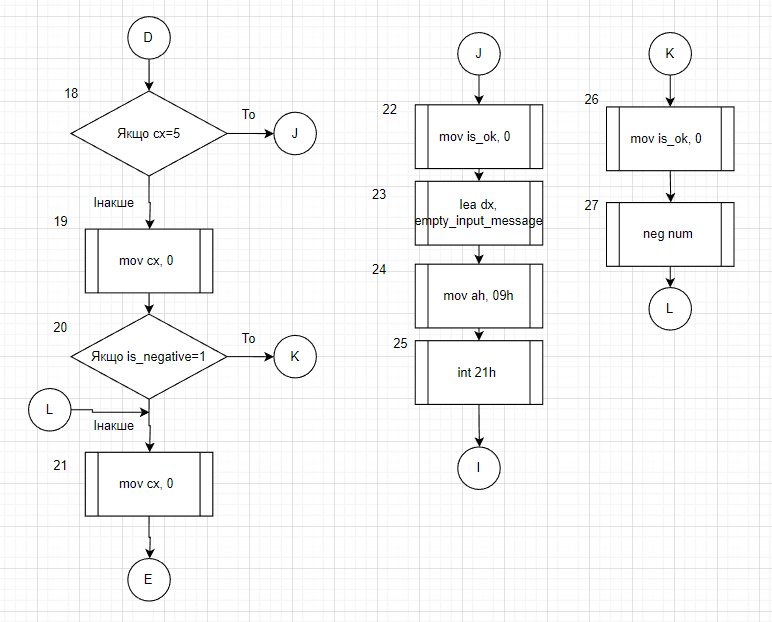
Блок-схема процедури function\_less\_0:

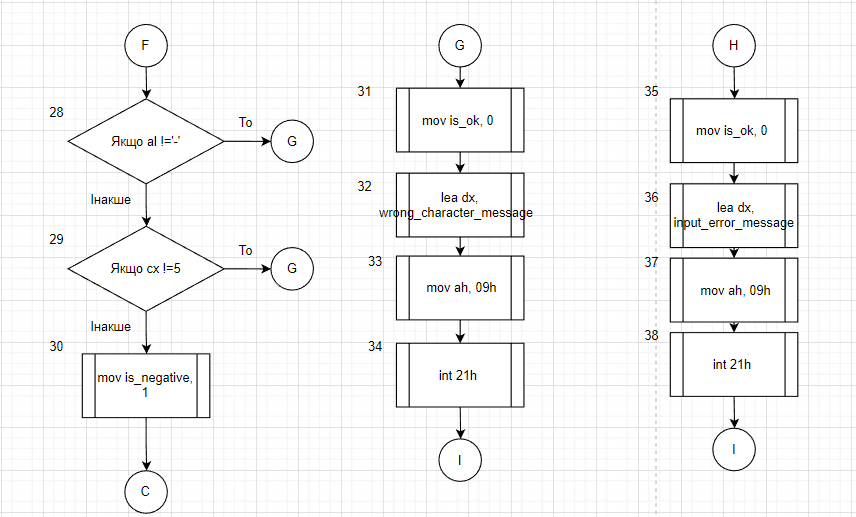


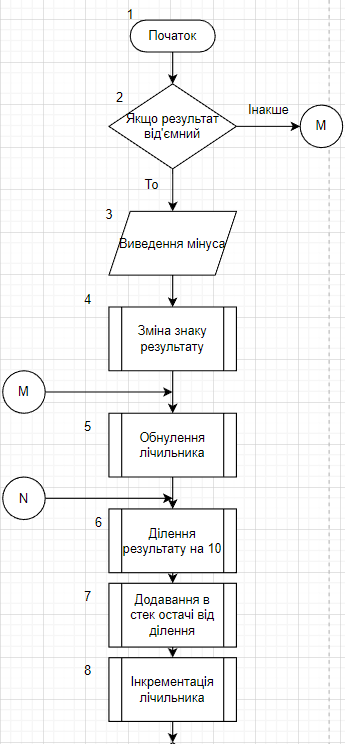
Блок-схема процедури read\_digit:

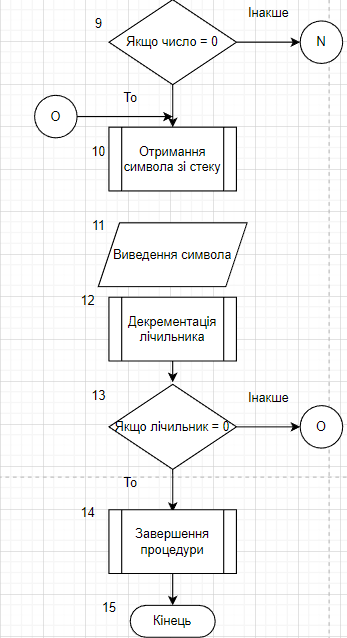




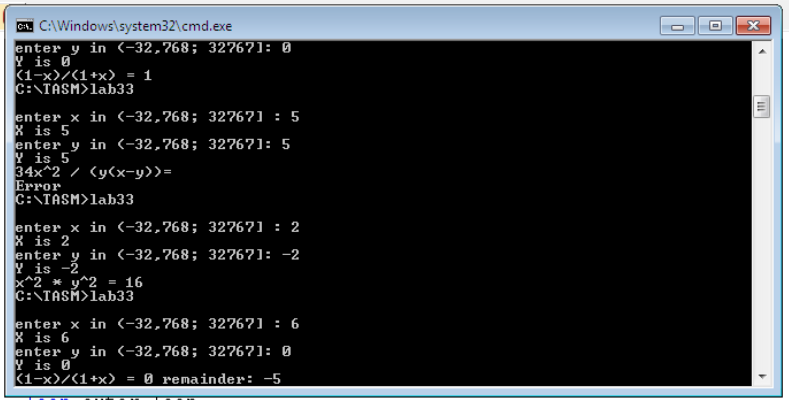




Блок-схема процедури print\_digit:  




**Тестування програми:**

  
  
Вручну:

Х = 5

Y = 5  
  
Випадок 1, але х=у, тому еррор  
--------------------------------------------------------------  
Х = 2

Y = -2  
  
Випадок 3: 2^2 \* (-2)^2 = 16

--------------------------------------------------------------

Х = 6

Y = 0  
  
Випадок 2: 1-6/1+6 - Ціла частина 0, остача 1-6= -5  
--------------------------------------------------------------  
  
Висновок: В текстовому редакторі написав програму за своїм варіантом завдання в файлі .asm. Програма обчислює значення заданої функції для змінних х та у та виводить результат. Побудував блок-схеми програми. Програма має захист від некоректно введених даних  
Таким чином, програма коректно виконує поставлену задачу.