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.  “...” ............... 2021 р. | Виконав  Студент групи ІП-01  Галько М.В. |

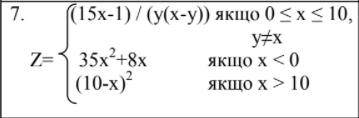
Київ – 2021

Комп’ютерний практикум No3

**Тема**: програмування розгалужених алгоритмів.

**Завдання**:

Написати програму, яка буде обчислювати значення функції.



**Текст програми:**

SSG SEGMENT PARA STACK 'STACK'

DB 64 DUP (?)

SSG ENDS

DSG SEGMENT PARA PUBLIC 'DATA'

MSG\_GET\_X DB 'X = $'

MSG\_GET\_Y DB 'Y = $'

MSG\_OVERFLOW DB 'OVERFLOW!$'

MSG\_WRONG\_DATA DB 'WRONG DATA!$'

MSG\_DIV\_ZERO DB 'ZERO DEVIDE!$'

X DW ?

Y DW ?

Z DW ?

Z\_SUB DW ?

ISNEG DB 0

BUFFER\_X DB 7, ?, 7 DUP(?)

BUFFER\_Y DB 7, ?, 7 DUP(?)

TEN DW 10

DSG ENDS

CSG SEGMENT PARA PUBLIC "CODE"

MAIN PROC FAR

ASSUME CS:CSG, DS:DSG, SS:SSG

PUSH DS

XOR AX, AX

PUSH AX

MOV AX, DSG

MOV DS, AX

CALL INPUT

CALL CHECKNUMS

CMP X, 0

JL FUNC2 ;35X^2 + 8X x < 0

CMP X, 10

JG FUNC3 ;(10-X)^2 x > 10

MOV AX, X

CMP AX, Y

JNE FUNC1 ;(15X-1)/(Y(X-Y)) 0 <= x <= 10 y!=x

JMP WRONG\_DATA

FUNC1: ;(15X-1)/(Y(X-Y))

MOV BX, Y

SUB AX, BX

JO OVERFLOW

IMUL BX ;Y(X-Y)

JO OVERFLOW

TEST AX, AX

JZ DIV\_ZERO

MOV Z, AX

MOV AX, X

MOV BX, 15

IMUL BX

JO OVERFLOW

SUB AX, 1 ;15X-1

JO OVERFLOW

XOR DX, DX

IDIV Z ;(15X-1)/(Y(X-Y))

MOV Z, AX

MOV Z\_SUB, DX

JMP FINISH

FUNC2: ;35X^2 + 8X

MOV AX, X

IMUL AX

JO OVERFLOW

MOV BX, 35

IMUL BX

JO OVERFLOW

MOV Z, AX ;35X^2

MOV AX, X

MOV BX, 8

IMUL BX ;8X

JO OVERFLOW

ADD Z, AX ;35X^2 + 8X

JO OVERFLOW

JMP FINISH

FUNC3: ;(10-X)^2

MOV AX, 10

SUB AX, X

JO OVERFLOW

IMUL AX

JO OVERFLOW

MOV Z, AX

JMP FINISH

WRONG\_DATA:

LEA DX, MSG\_WRONG\_DATA

JMP ERROR

OVERFLOW:

LEA DX, MSG\_OVERFLOW

JMP ERROR

DIV\_ZERO:

LEA DX, MSG\_DIV\_ZERO

JMP ERROR

FINISH:

CALL OUTPUT

RET

MAIN ENDP

CHECKNUMS PROC NEAR

XOR AX, AX

LEA SI, BUFFER\_X+1

MOV CL, [SI]

INC SI

CALL CHECK\_SIGN

CALL CHECK\_SYMBOLS

CMP ISNEG, 0

JE END\_CHECK\_X

NEG AX

END\_CHECK\_X:

MOV X, AX

MOV ISNEG, 0

XOR AX, AX

LEA SI, BUFFER\_Y+1

MOV CL, [SI]

INC SI

CALL CHECK\_SIGN

CALL CHECK\_SYMBOLS

CMP ISNEG, 0

JE END\_CHECK\_Y

NEG AX

END\_CHECK\_Y:

MOV Y, AX

RET

CHECK\_SIGN:

MOV BL, [SI]

CMP BL, '-'

JNE FINAL

INC SI

MOV ISNEG, 1

DEC CX

FINAL:

RET

CHECK\_SYMBOLS:

IMUL TEN

JO OVERFLOW

MOV BL, [SI]

CMP BL, '0'

JB WRONG\_DATA

CMP BL, '9'

JA WRONG\_DATA

SUB BL, '0'

ADD AX, BX

JNO INC\_FOR\_NEXT

CMP ISNEG, 0

JE OVERFLOW

CMP AX, 32768

JE OVERFLOW

INC\_FOR\_NEXT:

INC SI

LOOP CHECK\_SYMBOLS

RET

CHECKNUMS ENDP

ERROR PROC NEAR

MOV AH, 9

INT 21H

MOV AX, 4C00H

INT 21H

CALL NEXTLINE

ERROR ENDP

INPUT PROC NEAR

MOV AH, 9

LEA DX, MSG\_GET\_X

INT 21H

MOV AH, 10

LEA DX, BUFFER\_X

INT 21H

CALL NEXTLINE

MOV AH, 9

LEA DX, MSG\_GET\_Y

INT 21h

MOV AH, 10

LEA DX, BUFFER\_Y

INT 21H

CALL NEXTLINE

RET

NEXTLINE:

MOV AH, 2

MOV DL, 10 ; ASCII [10]/[AH] - NEW LINE

INT 21H

RET

INPUT ENDP

OUTPUT PROC NEAR

MOV AX, Z

TEST AX, AX

JNS SET\_BASIC

MOV AL, '-'

INT 29H

NEG Z

MOV AX, Z

SET\_BASIC:

XOR CX, CX

STACKING:

XOR DX, DX

DIV TEN

ADD DL, '0' ;'8' -> SCII CODE

PUSH DX

INC CX

TEST AX, AX

JNZ STACKING

UNSTACKING:

POP AX

INT 29H

LOOP UNSTACKING

MOV AX, Z\_SUB

TEST AX, AX

JZ ISZERO

STACKING\_SUB:

XOR DX, DX

DIV TEN

ADD DL, '0'

PUSH DX

INC CX

TEST AX, AX

JNZ STACKING\_SUB

MOV AL, '('

INT 29H

UNSTACKING\_SUB:

POP AX

INT 29H

LOOP UNSTACKING\_SUB

MOV AL, ')'

INT 29H

ISZERO:

CALL NEXTLINE

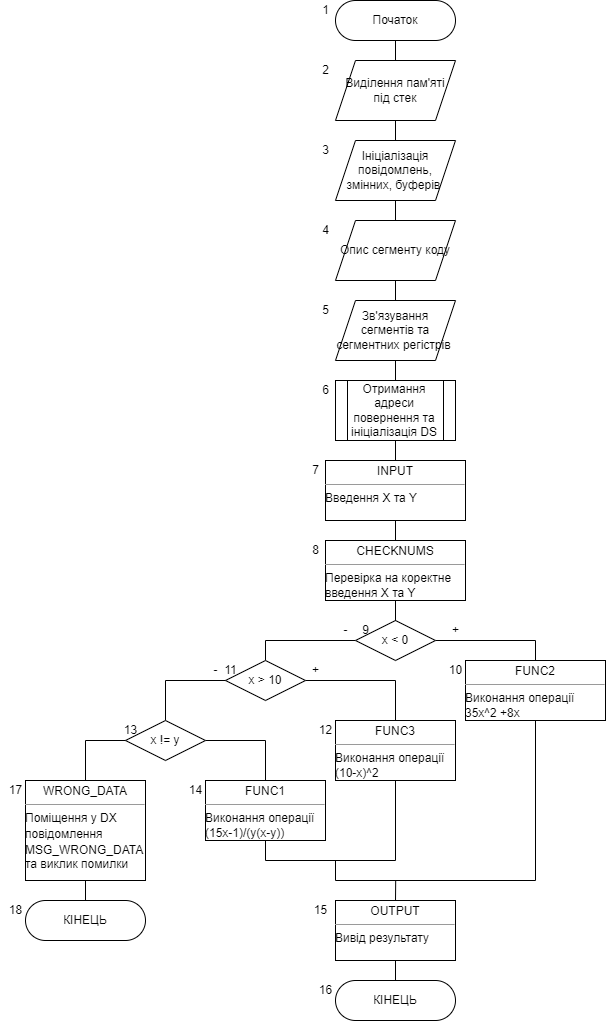
RET

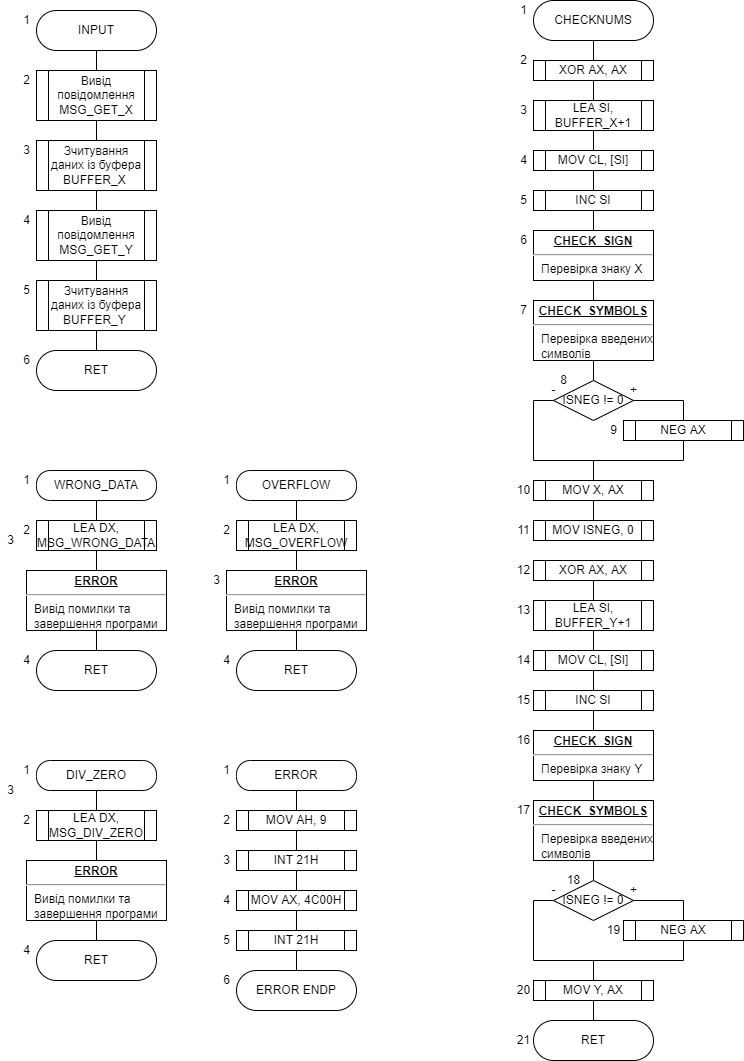
OUTPUT ENDP

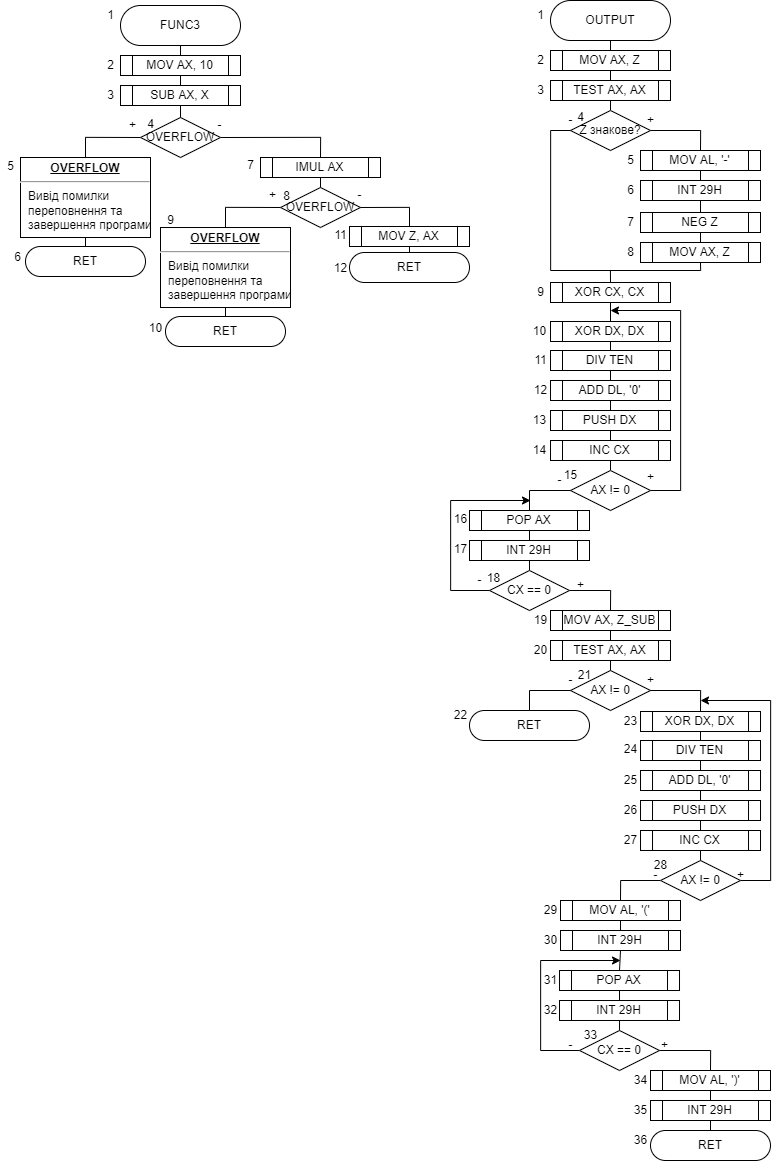
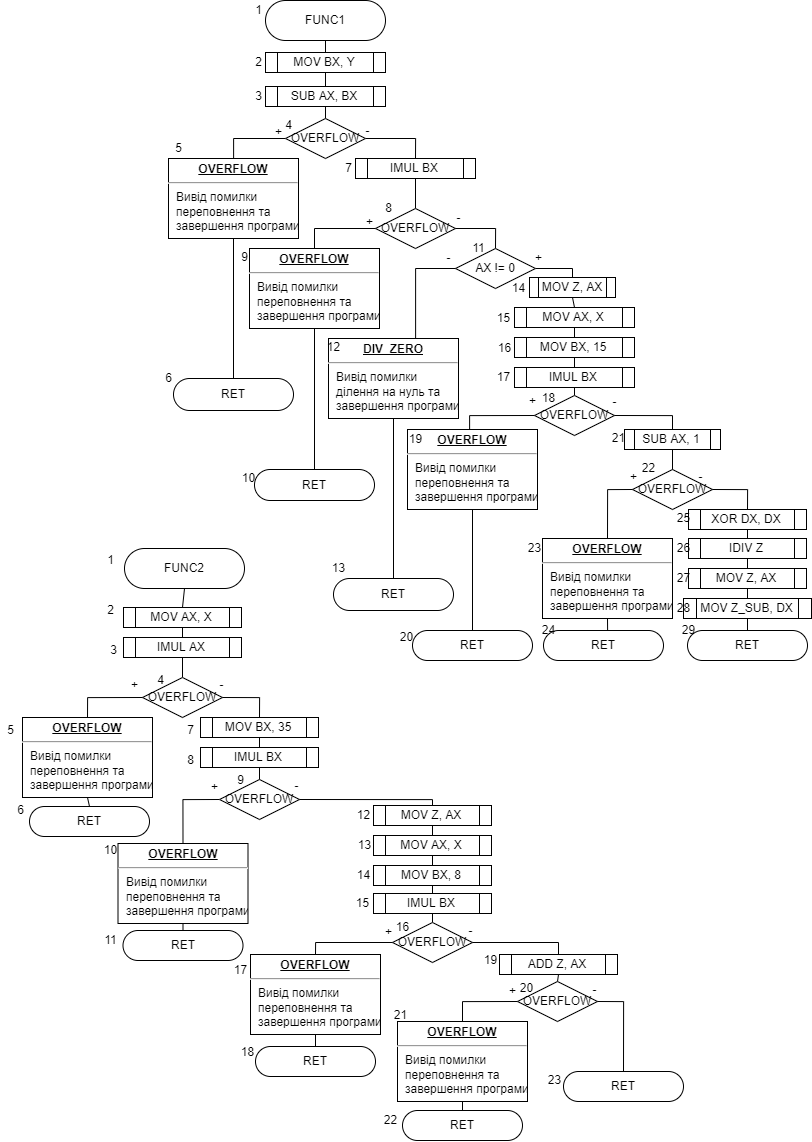
CSG ENDS

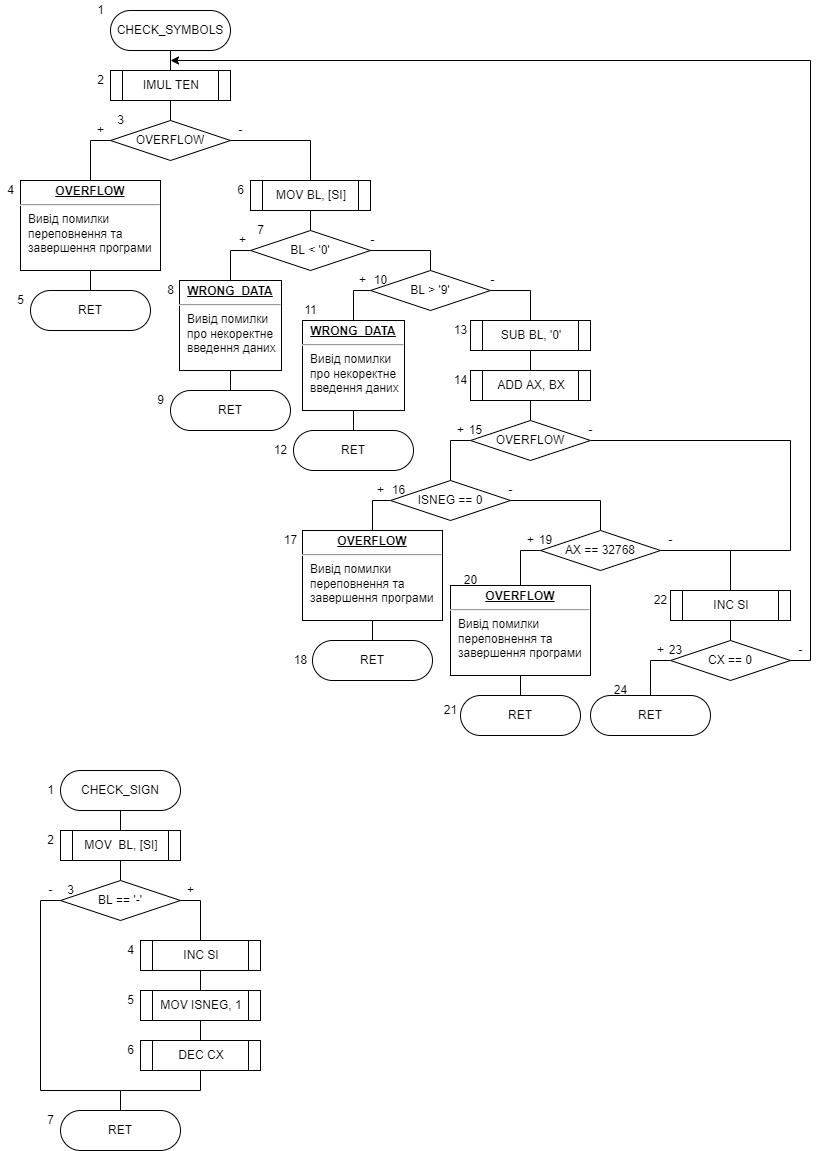
END MAIN

**Схема функціонування програми:**

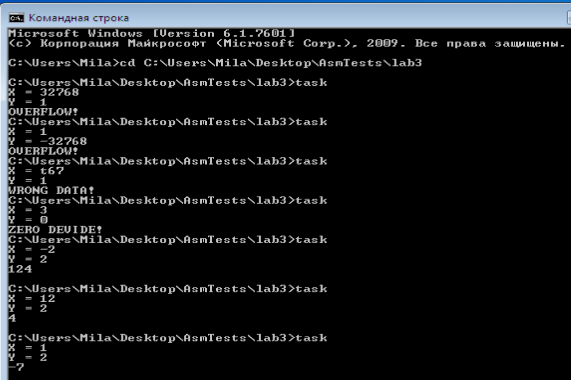
****

****

****



**Приклади роботи програми:**



**Висновок**:

В ході виконання комп’ютерного практикуму роботи я дослідила та навчилася програмувати розгалужені алгоритми. Спершу, була побудована блок-схема на основі якій був прописаний програмно хід алгоритму. Основними етапами алгоритму є перевірка на ввід коректних даних, виконання арифметичної дії в залежності від вхідних даних та вивід. Даний момент із вибором певної арифметичної дії вирішується використанням команд умовних переходів.

Програма реалізована правильно, судячи із кінцевих результатів тестів у пункті «Приклади роботи програми». Код програми також доданий до звіту (див. у пунктах «Текст програми» відповідно).