**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота № 7**

з дисципліни

«Системне програмування»

на тему

“Модульне програмування. Використання процедур.”

Виконав:

Перевірив:

студент групи ІП-93 доцент

Марченко Максим Олегович Павлов В.Г.

номер залікової книжки: 9316

Варіант 13

Київ 2021

Мета роботи : Вивчення прийомів модульного програмування, методів зверення до процедур і передачі в них параметрів.

Процедура - це обмежена частина програми, що виконує певну послідовність дій, і дозволяє звертатись до неї з будь-якого місця програми. На відміну від макросів, вони не вставляють свій код у місце виклику.

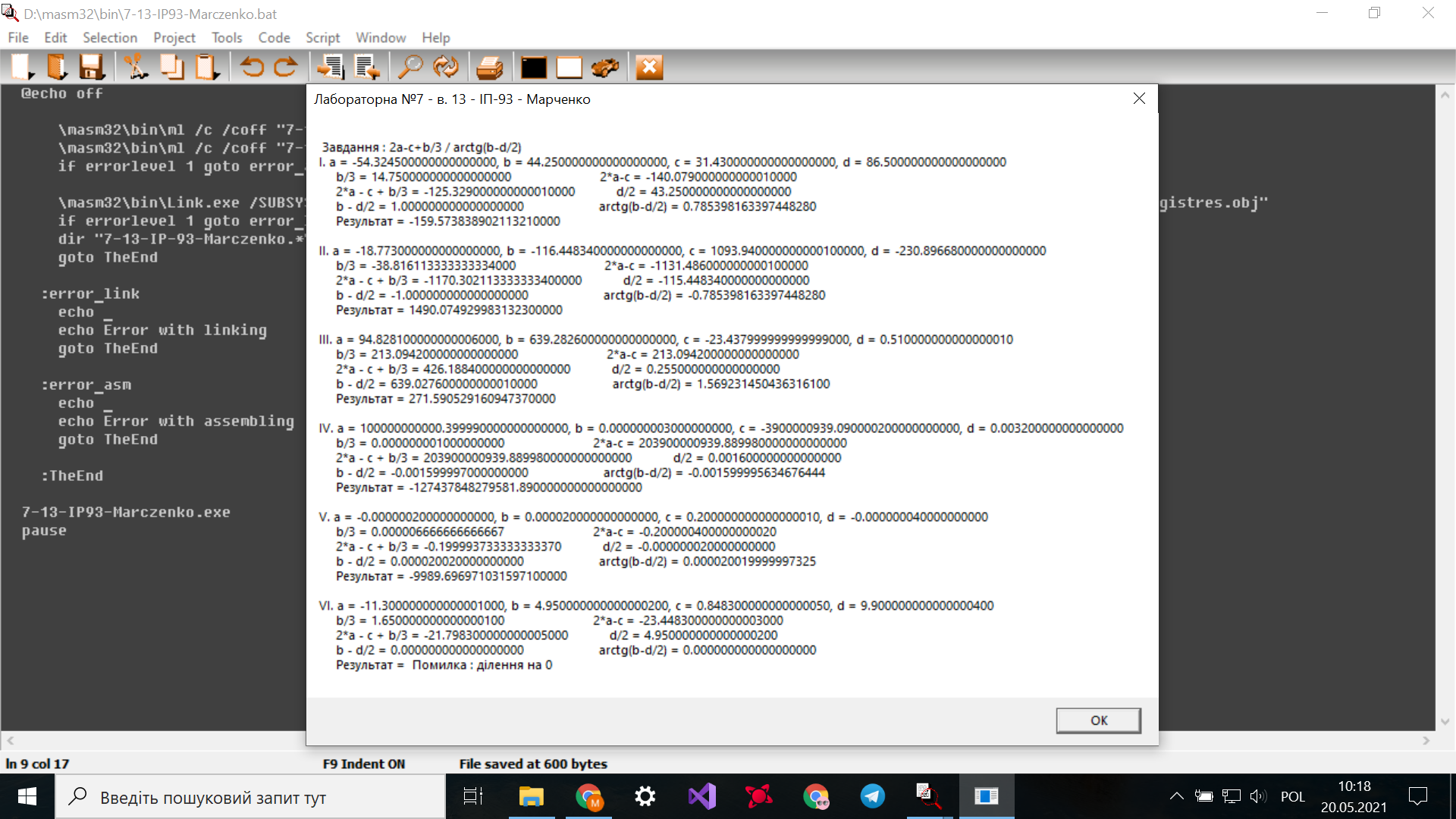
Процедура у асемблері обмежується ключовими словами PROC i ENDP, а також кожна процедура за допомогою команди RET повинна повертати певне значення (зазвичай встановлюється як 0, або результат, що вимагається).

Процедури повинні бути оголошені в тому ж файлі, де вони виконуються. Їх опис може бути і в іншому, підключеному файлі.

Є 3 основних методи передачі аргументів до процедур:

1. Користуючись з регістрів - процедура бере необхідні початкові аргументи з певних регістрів, виконує над ними певні дії, і результат записує у деякий регістр, звідки його може зчитати основна програма.
2. За допомогою стеку - основна програма заносить необхідні аргументи в стек і передає процедурі, що виносить окремі аргументи з нього, вписує результат в стек, і передає тілу програми.
3. За допомогою extern-public - у основній програмі створюються змінні, позначені директивою екстерн - аргументи процедури, та паблік - результат виводу процедури. В самій процедурі також створюються змінні екстерн-типу, а результат виводу функції автоматично записується в змінну паблік.

Скріншот виконання програми



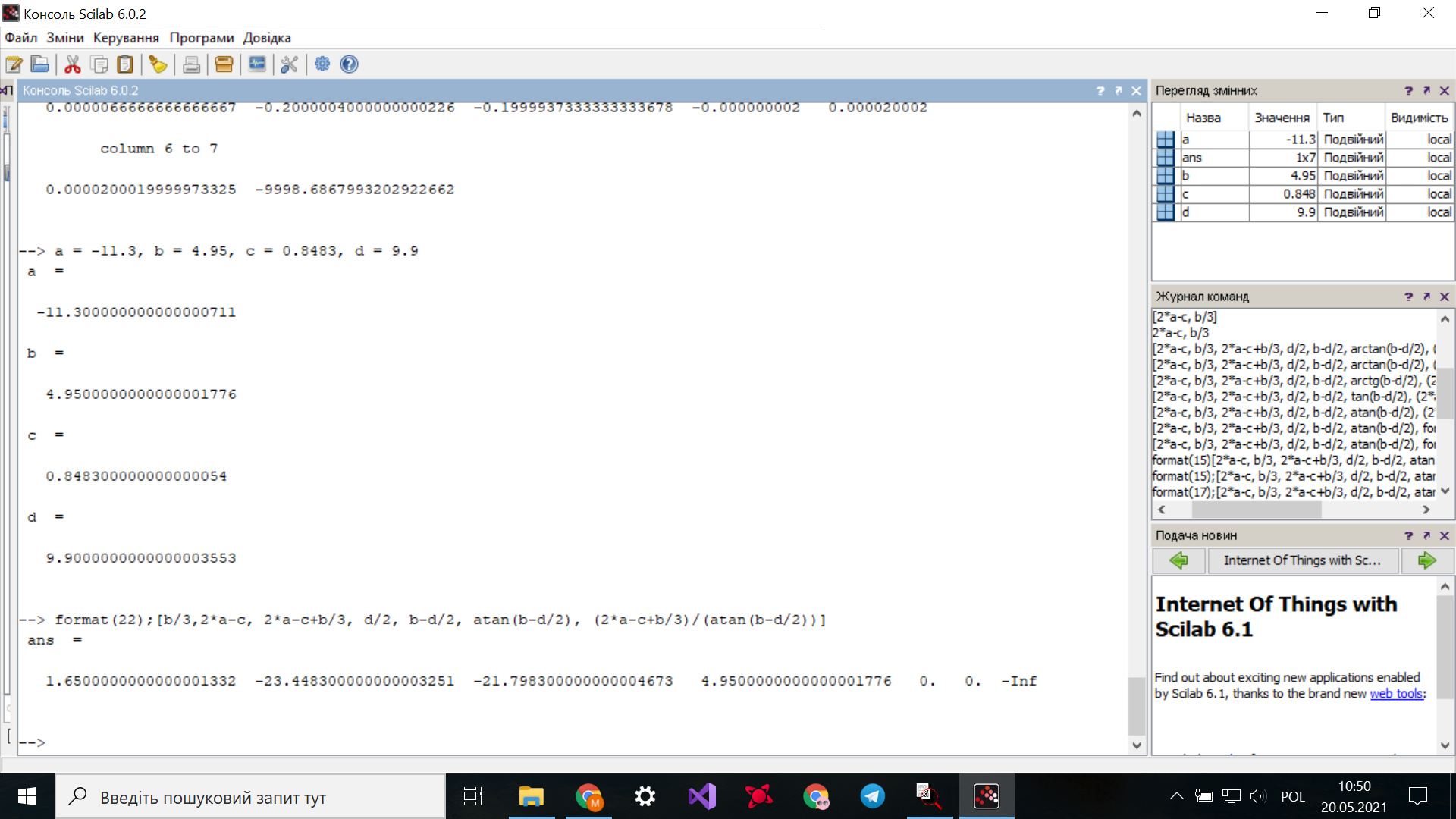
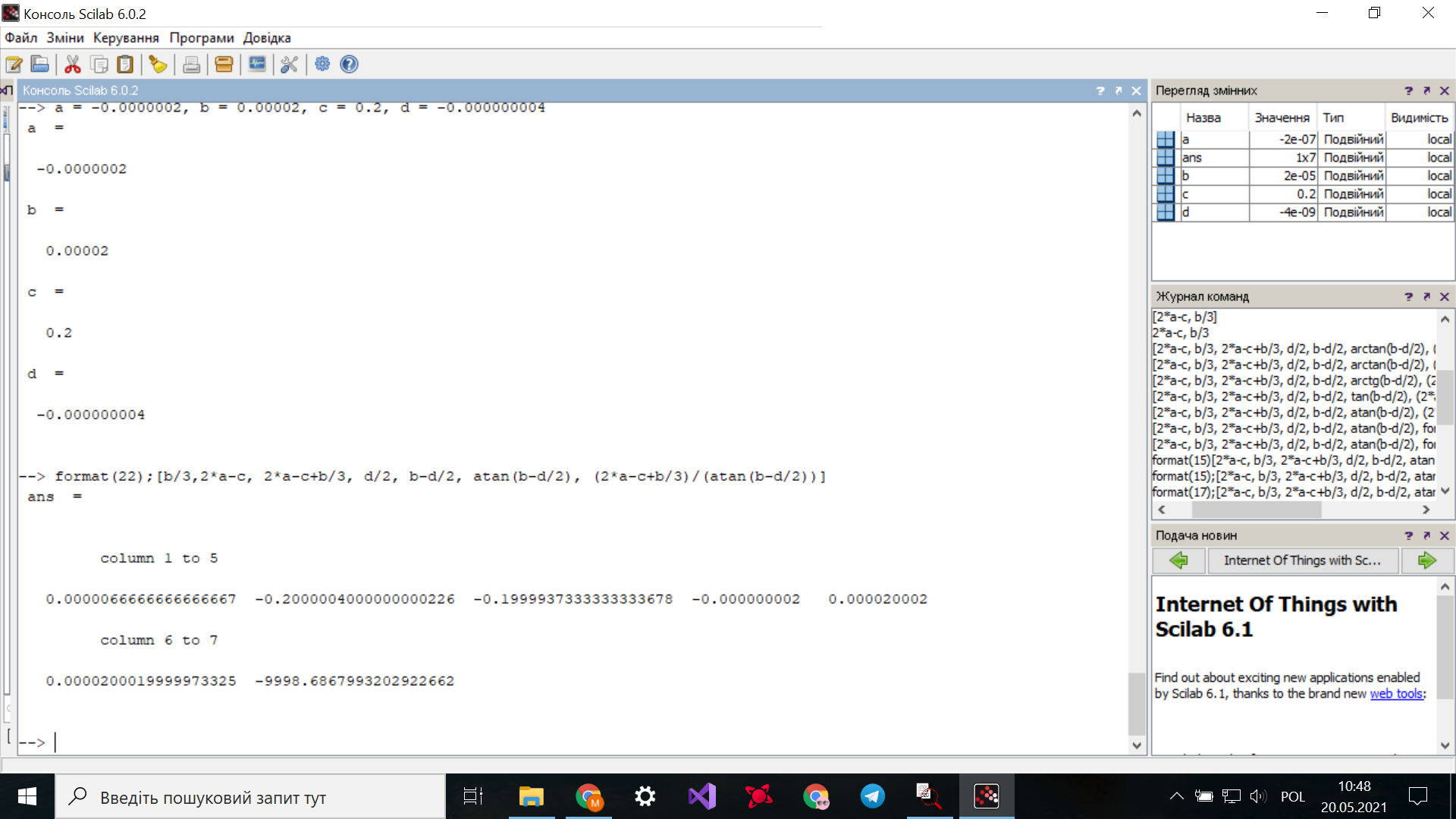
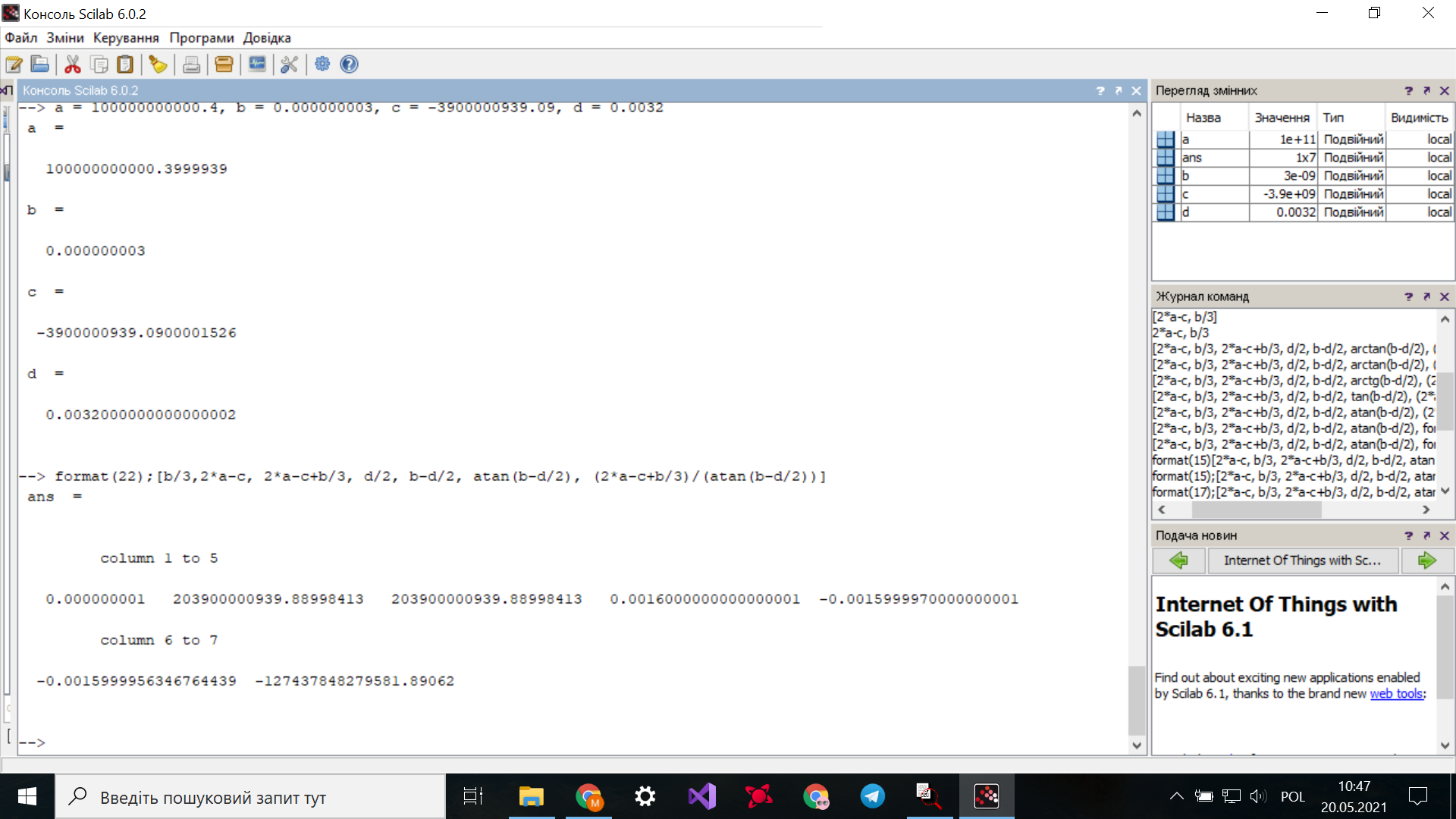
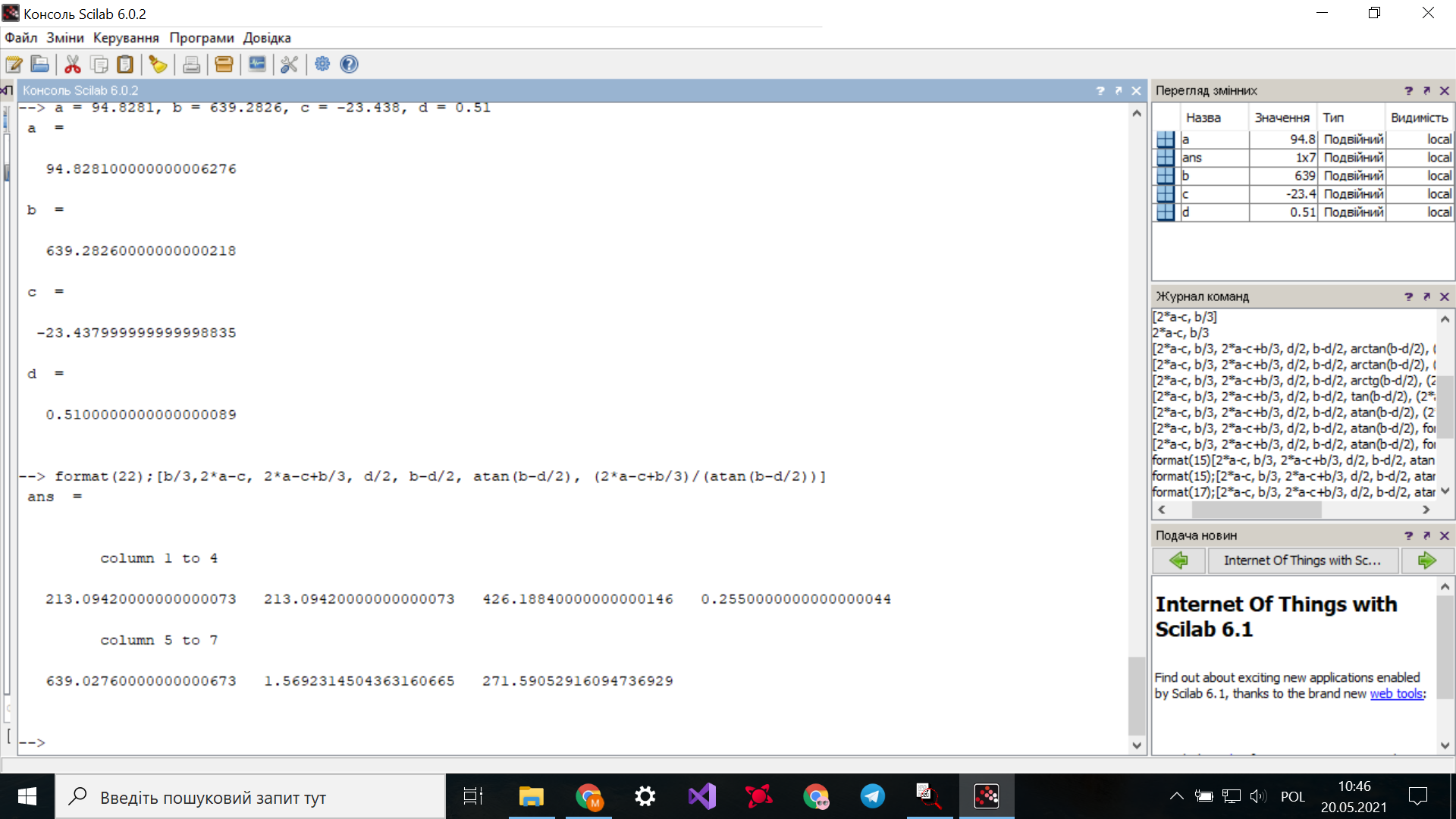
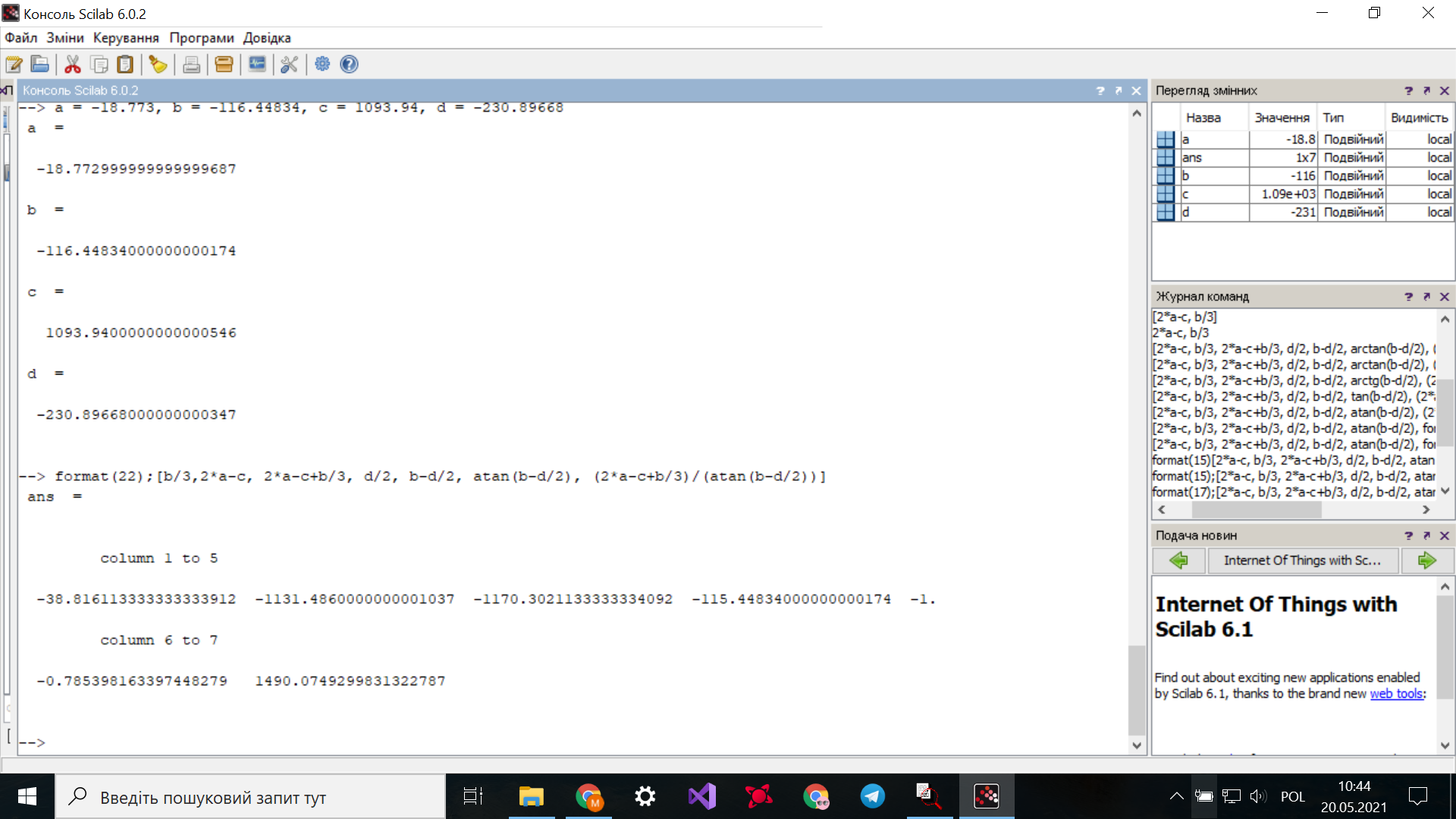
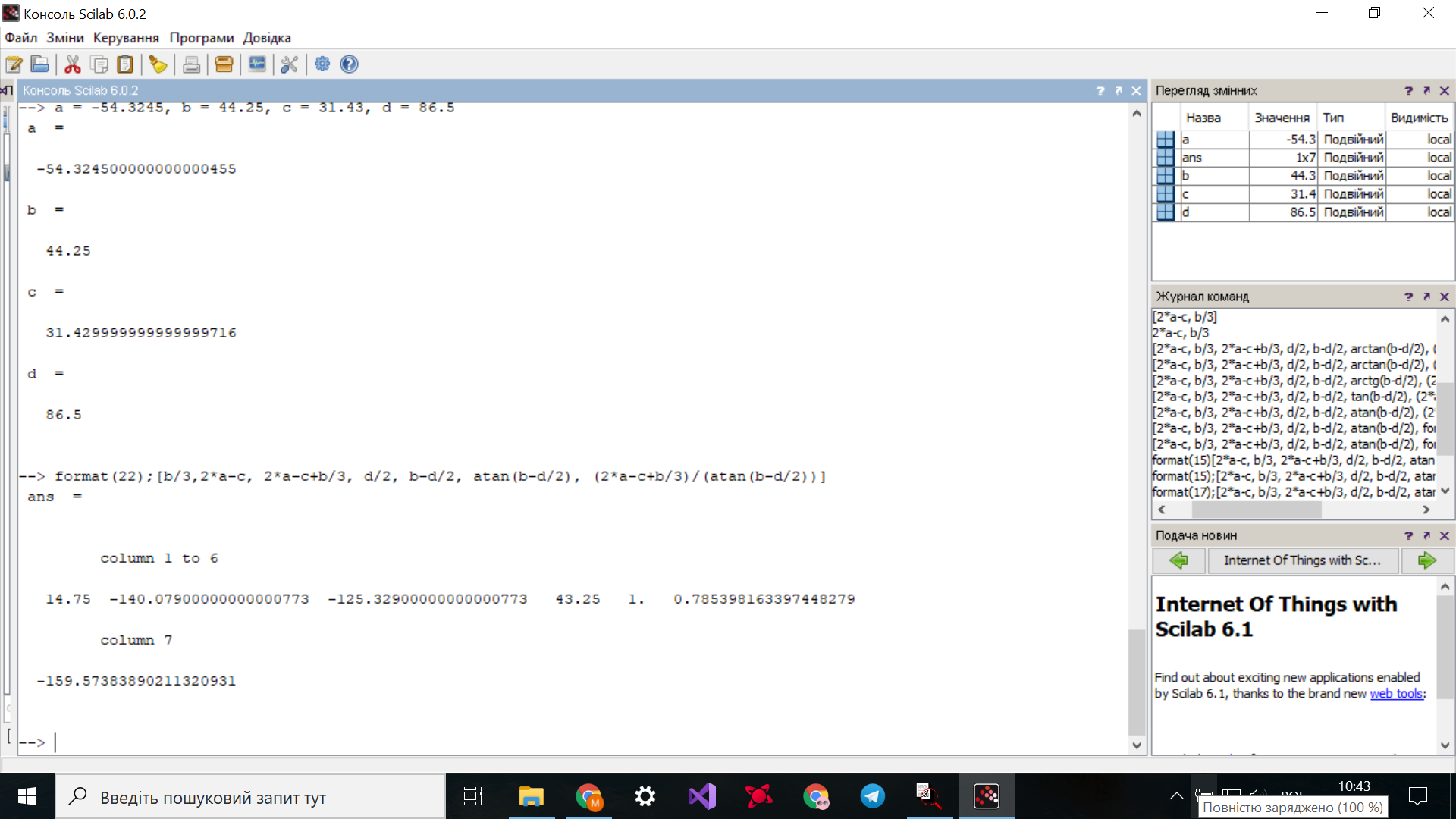
На скріншоті зображено виконання програми з 6 наборами вхідних даних, 5 з яких - коректні, 1 - спричинює ділення на 0.

Для знаходження помилки ділення на 0 знаменник виразу порівнюється з нулем перед фінальною дією ділення чисельник на знаменник. Якщо знаменник рівний нулю, то ця дія не виконується, замість її результату виводиться повідомлення про помилку.

Контрольні розрахунки

Розрахунки проводитимуться в тому ж порядку, як і в програмі. Черговість дій:

1. b/3
2. 2a-c
3. 2a-c+b/3
4. d/2
5. b-d/2
6. arctg(b-d/2)
7. (2a-c+b/3) / arctg(b-d/2)



Дії для контрольних розрахунків проведені в тій ж послідовності, що й в програмі.

Як видно, результат виконання програми програми та контрольні розрахунки збігаються з точністю до заокруглення для всіх вхідних даних, включаючи випадки помилок, що підтверджує правильність програми. Похибка у 5) прикладі пояснюється тим, що у ньому використовуються дуже малі числа, і тому будь-яка похибка заокруглення результату при переведенні результату з двійкового в десятковий режим (що спостерігається як у масм, так і в прикладах перевірки) створює значний вплив на підрахунок фінального результату.

Висновки

При виконанні роботи я навчився використовувати процедури у своїх програмах, що забезпечує її модульність, тобто полегшує можливість використання певних процедур в інших проєктах. Існує 3 основних способи передачі аргументів - через регістри, за допомогою стеку та через директиви extern-public. Перший метод є найменш зручним, оскільки має дуже обмежену к-ть аргументів та вимагає передачі правильних аргументів у правильні регістри кожного разу при виклику процедури, і тому рідко використовується на практиці. Варіант передачі даних через стек є найпоширенішим, оскільки не має таких обмежень, як перший метод, та є легким у використанні. Часто зустрічається також і метод директив, оскільки відносно легко масштабується.

Додаток: Код програми 7-13-IP-93-Marczenko

include \masm32\include\masm32rt.inc

include \masm32\include\Fpu.inc

includelib \masm32\lib\Fpu.lib

extern operation1:PROTO

extern oper3:PROTO

extern out\_results: qword

extern operation2:PROTO

public in\_b, in\_d

write\_in MACRO buff, num, error

local out\_if

local end\_if

cmp error, 0

je out\_if

invoke crt\_sprintf, addr buff, addr error\_message

jmp end\_if

out\_if:

invoke crt\_sprintf, addr buff, addr number\_message, num

end\_if:

ENDM

move\_for8 MACRO in, out

local cycle1

mov ax, 0

cycle1:

mov esi, DWORD ptr in[0]

mov DWORD ptr out[0], esi

inc ax

cmp ax, 2

;jne cycle1

mov esi, DWORD ptr in[4]

mov DWORD ptr out[4], esi

cmp ax, 2

;jne cycle1

ENDM

enumerate MACRO a, b, cc, d, res

mov is\_error, 0

; operation 1 - start

lea esi, a

lea edx, cc

lea eax, count\_first

call operation1

mov is\_error, 0

write\_in buff2, count\_first, is\_error

; operation 1 - end

; operation 2 - start

push dword ptr b[4]

push dword ptr b[0]

push offset count\_second

call operation2

write\_in buff1, count\_second, is\_error

; operation 2 - end

; operation 3 - start

move\_for8 b, in\_b

move\_for8 d, in\_d

call oper3

;cmp out\_results, 0

write\_in buff4, out\_results[16], is\_error

write\_in buff5, out\_results[8], is\_error

write\_in buff6, out\_results, is\_error

; operation 3 - end

finit

fld count\_first

fld count\_second

fadd st, st(1)

fstp count\_first

write\_in buff3, count\_first, is\_error

fld count\_first

fld out\_results

fcom nulle

fstsw ax

sahf

jz error\_is

fdivp st(1), st

fstp res

jmp out\_if

error\_is:

mov eax, 1

mov is\_error, eax

out\_if:

write\_in buff7, res, is\_error

ENDM

calculate MACRO i, buff

enumerate a[i\*8], b[i\*8], cc[i\*8], d[i\*8], res[i\*8]

invoke crt\_sprintf, buff, addr out\_message, a[i\*8], b[i\*8], cc[i\*8], d[i\*8],

addr buff1, addr buff2, addr buff3, addr buff4,

addr buff5, addr buff6, addr buff7

ENDM

.data

header DB " Лабораторна №7 - в. 13 - ІП-93 - Марченко", 0

template DB " Завдання : 2a-c+b/3 / arctg(b-d/2)", 10,

"I. %s", 10,

"II. %s", 10,

"III. %s", 10,

"IV. %s", 10,

"V. %s", 10,

"VI. %s", 0

out\_message DB "a = %.18f, b = %.18f, c = %.18f, d = %.18f", 10,

" b/3 = %s"," 2\*a-c = %s",10,

" 2\*a - c + b/3 = %s"," d/2 = %s",10,

" b - d/2 = %s"," arctg(b-d/2) = %s",10,

" Результат = %s",10, 0

error\_message DB " Помилка : ділення на 0", 0

buff1 DB 128 DUP (0)

buff2 DB 128 DUP (0)

buff3 DB 128 DUP (0)

buff4 DB 128 DUP (0)

buff5 DB 128 DUP (0)

buff6 DB 128 DUP (0)

buff7 DB 128 DUP (0)

buffer\_now DD 0

out\_buffer DB 4096 DUP (0)

buffer\_ex1 DB 1024 DUP (0)

buffer\_ex2 DB 1024 DUP (0)

buffer\_ex3 DB 1024 DUP (0)

buffer\_ex4 DB 1024 DUP (0)

buffer\_ex5 DB 1024 DUP (0)

buffer\_ex6 DB 1024 DUP (0)

buff\_size = $ - buffer\_ex6

res DQ 6 DUP (0)

a DQ -54.3245, -18.773, 94.8281, 100000000000.4, -0.0000002, -11.3

b DQ 44.25, -116.44834, 639.2826 , 0.000000003 , 0.00002, 4.95

cc DQ 31.43, 1093.94, -23.438, -3900000939.09 , 0.2, 0.8483

d DQ 86.5, -230.89668, 0.51 , 0.0032 , -0.00000004, 9.9

in\_b DQ 0.0

in\_d DQ 0.0

count\_first DQ 0.0

count\_second DQ 0.0

count\_third DQ 0.0

is\_error DD 0

nulle DQ 0.0

number\_message DB "%.18f", 0

.code

start:

mov edi, 0

mov buffer\_now, offset buffer\_ex1

program\_cycle:

calculate edi, buffer\_now

add buffer\_now, buff\_size

inc edi

cmp edi, 6

jb program\_cycle

invoke crt\_sprintf, addr out\_buffer, addr template,

addr buffer\_ex1, addr buffer\_ex2,

addr buffer\_ex3, addr buffer\_ex4,

addr buffer\_ex5, addr buffer\_ex6

invoke MessageBox, 0, addr out\_buffer, addr header, MB\_OK

invoke ExitProcess, 0

END start