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-21**

**Лісовиченко О.І. Скрипець О.О.**

**“06” травня 2024 р.**

Київ – 2024

**Комп’ютерний практикум №3**

**Тема:** програмування розгалужених алгоритмів.

**Завдання**:

Написати програму, яка повинна мати наступний функціонал:

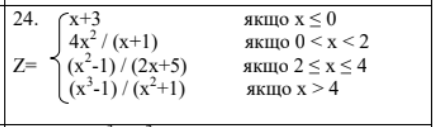
1. Можливість введення користувачем значень x, y, t, a, b за необхідності.

2. Обчислювати значення функції за введеними значеннями.

3. Виводити на екран результат обчислень.

4. Якщо є ділення, то результат дозволяється виводити: окремо цілу частину та остачу

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



Текст програми

sseg *segment* para stack 'stack'

         db 64 dup ( 'STACK' )

sseg *ends*

dseg *segment* para public 'data'

    num              dw 0

    chuselnyk        dw 0

    znamennyk        dw 0

    temp             dw 0

    ostacha          dw 0

    x                dw 0

    result           dw 0

    is\_minus         dw 0

    enter\_x          db 0dh, 0ah, 'enter X in range of -32767 to 32767: : $'

    print\_ostacha    db ' ostacha: $'

    overflow\_error   db 'Overflow $'

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

    wrong\_char\_msg   db 0dh, 0ah, 'wrong character$'

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

    less\_or\_equals\_0 db 0dh, 0ah, 'x+3 = $'

    between\_0\_and\_2  db 0dh, 0ah, '4x^2/(x+1) = $'

    between\_2\_and\_4  db 0dh, 0ah, '(x^2-1)/(2x+5)= $'

    greater\_than\_4   db 0dh, 0ah, '(x^3-1)/(x^2+1)= $'

dseg *ends*

cseg *segment* para public 'code'

main *proc*

    main\_loop:

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

                              push   ds

                              xor    ax, ax

                              push   ax

                              mov    ax, dseg

                              mov    ds, ax

                              lea    dx, enter\_x

                              mov    ah, 09h

                              int    21h

                              call   read\_digit

                              mov    ax,num

                              mov    x, ax

                              cmp    x, 0

                              jle    func\_less\_or\_equals\_0           ;менше дорівнює 0

                              cmp    x, 2

                              jl     func\_between\_0\_and\_2            ;між 0 і 2

                              cmp    x, 4

                              jle    func\_between\_2\_and\_4            ;між 2 і 4

                              jmp    func\_greater\_4                  ;більше 4

    ; менше

    func\_less\_or\_equals\_0:

                              lea    dx, less\_or\_equals\_0

                              mov    ah, 09h

                              int    21h

                              call   function\_less\_or\_equals\_0

                              jmp    main\_loop

    func\_between\_0\_and\_2:

                              lea    dx, between\_0\_and\_2

                              mov    ah, 09h

                              int    21h

                              call   function\_between\_0\_and\_2

                              jmp    main\_loop

    func\_between\_2\_and\_4:

                              lea    dx, between\_2\_and\_4

                              mov    ah, 09h

                              int    21h

                              call   function\_between\_2\_and\_4

                              jmp    main\_loop

    func\_greater\_4:

                              lea    dx, greater\_than\_4

                              mov    ah,09h

                              int    21h

                              call   function\_greater\_than\_4

                              jmp    main\_loop

main *endp*

function\_less\_or\_equals\_0 *proc*

                              xor    ax,ax

                              xor    cx,cx                           ;х+3

                              mov    ax, x

                              add    ax, 3

                              jo     func1\_overflow

                              MOV    result, ax

                              call   print\_result

                              ret

    func1\_overflow:

                              lea    dx, overflow\_error

                              mov    ah, 09h

                              int    21h

                              ret

function\_less\_or\_equals\_0 *endp*

function\_between\_0\_and\_2 *proc*

                              xor    ax,ax

                              xor    cx,cx

                              mov    ax, x

                              mov    cx, x

                              imul   cx                              ;ax= x^2

                              jc     func2\_overflow

                              jo     func2\_overflow

                              mov    result,ax                       ; res = x^2

                              xor    ax,ax

                              MOV    ax,4

                              imul   result                          ; ax=4x^2

                              jc     func2\_overflow

                              jo     func2\_overflow

                              mov    result,ax                       ; res = 34x^2

                              jo     func2\_overflow

                              mov    chuselnyk, ax

                              mov    ax, x

                              add    ax, 1                           ; x+1

                              mov    znamennyk, ax

                              mov    ax, chuselnyk

                              mov    bx, znamennyk

                              cwd

                              idiv   bx                              ; 4x^2/(x+1)

                              MOV    result, AX

                              mov    ostacha, dx

                              call   print\_result

                              cmp    ostacha,0

                              jne    func2\_ostacha                   ; якщо результат не дорівнює 0 перехід сюди

                              ret

    func2\_ostacha:

                              lea    dx, print\_ostacha

                              mov    ah, 09h

                              int    21h

                              call   print\_num\_ostacha

                              ret

    func2\_overflow:

                              lea    dx, overflow\_error

                              mov    ah, 09h

                              int    21h

                              ret

function\_between\_0\_and\_2 *endp*

function\_between\_2\_and\_4 *proc*

                              xor    ax,ax

                              xor    cx,cx

                              mov    ax, x

                              mov    cx, x

                              imul   cx                              ;ax= x^2

                              sub    ax, 1                           ; x^2 - 1

                              jc     func3\_overflow

                              jo     func3\_overflow

                              mov    chuselnyk, ax

                              mov    ax, x

                              add    ax, ax                          ; 2x

                              add    ax, 5                           ; 2x + 5

                              jc     func3\_overflow

                              jo     func3\_overflow

                              mov    znamennyk, ax

                              mov    ax, chuselnyk

                              cwd

                              idiv   znamennyk                       ; (x^2 - 1) / (2x + 5)

                              MOV    result, AX

                              mov    ostacha, dx

                              call   print\_result

                              cmp    ostacha,0

                              jne    func3\_ostacha

                              ret

    func3\_ostacha:

                              lea    dx, print\_ostacha

                              mov    ah, 09h

                              int    21h

                              call   print\_num\_ostacha

                              ret

    func3\_overflow:

                              lea    dx, overflow\_error

                              mov    ah, 09h

                              int    21h

                              ret

function\_between\_2\_and\_4 *endp*

function\_greater\_than\_4 *proc*

                              xor    ax,ax

                              xor    cx,cx

                              xor    dx,dx

                              mov    ax, x

                              mov    cx, x

                              imul   cx                              ;ax= x^2

                              mov    bx, x

                              imul   bx                              ; ax=x^3

                              sub    ax, 1                           ; x^3 - 1

                              jc     func4\_overflow

                              jo     func4\_overflow

                              mov    chuselnyk, ax                   ;n=x^3 - 1

                              xor    ax,ax

                              xor    cx,cx

                              xor    dx,dx

                              mov    ax, x

                              mov    cx, x

                              imul   cx                              ;ax= x^2

                              add    ax, 1                           ; x^2 + 1

                              jc     func4\_overflow

                              jo     func4\_overflow

                              mov    znamennyk, ax                   ;den = x^2 + 1

                              mov    ax, chuselnyk

                              cwd

                              idiv   znamennyk                       ; (x^3 - 1) / (x^2 + 1)

                              mov    result, ax

                              mov    ostacha, dx

                              call   print\_result

                              cmp    ostacha,0

                              jne    func4\_ostacha

                              ret

    func4\_ostacha:

                              lea    dx, print\_ostacha

                              mov    ah, 09h

                              int    21h

                              call   print\_num\_ostacha

                              ret

    func4\_overflow:

                              lea    dx, overflow\_error

                              mov    ah, 09h

                              int    21h

                              ret

function\_greater\_than\_4 *endp*

read\_digit *proc*

                              mov    num,0

                              mov    bx, 10

                              mov    cx, 5

                              mov    is\_minus,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\_minus, 1

                              je     make\_negative

                              ret

    stop:

                              cmp    cx, 5

                              je     empty\_input

                              mov    cx, 0

                              cmp    is\_minus, 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\_minus, 1

                              jmp    read

    input\_error:

                              lea    dx, input\_error\_msg

                              mov    ah, 09h

                              int    21h

                              jmp    main\_loop

    wrong\_character:

                              lea    dx, wrong\_char\_msg

                              mov    ah, 09h

                              int    21h

                              jmp    main\_loop

    empty\_input:

                              lea    dx, empty\_input\_msg

                              mov    ah, 09h

                              int    21h

                              jmp    main\_loop

read\_digit *endp*

print\_num\_ostacha *proc*

                              mov    bx, ostacha

                              or     bx, bx

                              jns    positive\_ost

                              mov    al, '-'

                              int    29h

                              neg    bx

    positive\_ost:

                              mov    ax, bx

                              xor    cx, cx

                              mov    bx, 10

    transform\_ost:

                              xor    dx, dx

                              div    bx

                              add    dl, '0'

                              push   dx

                              inc    cx

                              test   ax, ax

                              jnz    transform\_ost

    print\_ost:

                              pop    ax

                              int    29h

                              loop   print\_ost

                              ret

print\_num\_ostacha *endp*

print\_result *proc*

                              mov    bx, result

                              or     bx, bx

                              jns    positive\_res

                              mov    al, '-'

                              int    29h

                              neg    bx

    positive\_res:

                              mov    ax, bx

                              xor    cx, cx

                              mov    bx, 10

    transform\_res:

                              xor    dx, dx

                              div    bx

                              add    dl, '0'

                              push   dx

                              inc    cx

                              test   ax, ax

                              jnz    transform\_res

    print\_res:

                              pop    ax

                              int    29h

                              loop   print\_res

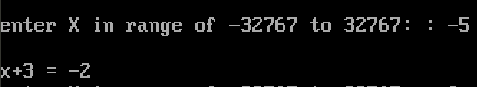
                              ret

print\_result *endp*

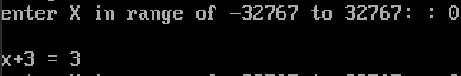
cseg *ends*

    end main

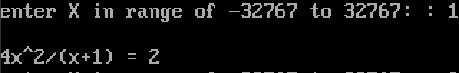
**Перевірки:**

****

**Х<0, тому -5+3=-2**

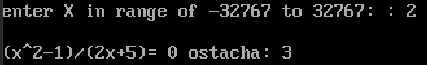
****

**Х=0, тому 0+3=3**

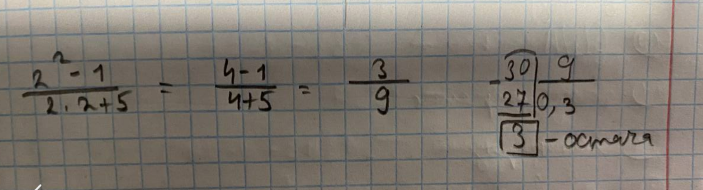
****

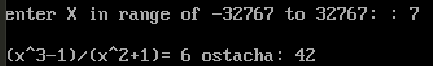
**0<Х<2, тому**

****

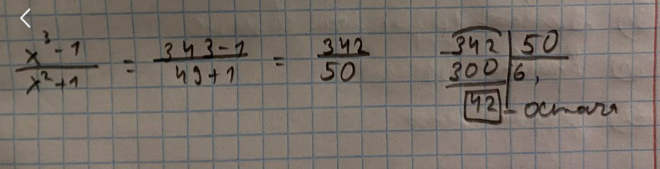


2<=X <=4, тому





Х>4, тому



**Висновок:**

У цій лабораторній роботі ми розглянули створення програми, яка включає в себе введення даних користувачем, обчислення значень функції, виведення результатів на екран, а також обробку помилок вводу.

Ця робота також підкреслила важливість ретельного тестування та налагодження коду, щоб забезпечити його коректну роботу в усіх можливих сценаріях використання. Загалом, ця лабораторна робота була корисною для поглиблення наших знань про програмування та розуміння того, як можна ефективно структурувати та організувати код.

Ця робота також допомогла нам краще зрозуміти, як можна використовувати програмування для вирішення реальних проблем та виконання реальних завдань, що є важливим аспектом навчання програмуванню.