Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1
«Основные конструкции языка Python (исполнение на fasm)»

Выполнил:
студент группы ИУ5-35Б
Никулин И.М.
Подпись и дата:

Проверил:
преподаватель
Гапанюк Ю. В.
Подпись и дата:

Москва, 2024 г

Постановка задачи

Разработать программу для решения биквадратного уравнения.

Программа должна быть разработана в виде консольного приложения на языке Python.
Программа осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
Коэффициенты A, B, C могут быть заданы в виде параметров командной строки (вариант задания параметров приведен в конце файла с примером кода). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.
Если коэффициент A, B, C введен или задан в командной строке некорректно, то необходимо проигнорировать некорректное значение и вводить коэффициент повторно пока коэффициент не будет введен корректно.
Корректно заданный коэффициент - это коэффициент, значение которого может быть без ошибок преобразовано в действительное число.
Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
Дополнительное задание 2 (*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме C++).

Текст программы

```
format ELF64


public _start


extrn print_float

extrn new_line


section '.bss' writable

    msg_neg_D db "Нет корней", 0

    msg_zero_D db "Два корня", 0

    msg_roots db "Корни уровнения", 0


    buffer dq 0.0
```

```
    n dq 0
    four dq 4.0
    two dq 2.0
    zero dq 0.0

    ; ВВОД
    a dq 3.0
    b dq 4.0
    c dq -5.0


section '.text' executable

_start:
    fld qword [four] ; 7
    fld qword [two] ; 6
    fld qword [n] ; 5
    fld qword [n] ; 4
    fld qword [b] ; 3
    fld qword [c] ; 2
    fld qword [a] ; 1
    fld qword [b] ; 0

    fmul st0, st0     ; b * b

    fxch
    fmul st0, st1     ; a * c
    fmul st0, st7     ; 4ac
    fsub st2, st0     ; b * b - 4ac
    fxch st2
```

```
fxch st1
fmul st0, st6     ; 2a
fxch st1

fld qword [zero]
fcomp
jl neg_D ; D < 0

fsqrt

fst st5
fxch st5
fsub st0, st4 ; -b + sqrt d
fxch st5
fadd st0, st4 ; b + sqrt d
fxch st5
fdiv st0, st1 ; x1
fxch st5      ; x1 - st5
fdiv st0, st1 ; x2

je zeroD ; D = 0

mov rsi, msg_roots
call print_str
call new_line

fld qword [zero]
fcomp ; x2 ? 0
jl neg_root
je .one_root
```

```
fsqrt
fchs
fstp [buffer]
mov rax, [buffer]
push rax
call print_float
add rsp, 8
call new_line
call exit
fchs

.one_root:
    fstp [buffer]
    mov rax, [buffer]
    push rax
    call print_float
    add rsp, 8
    call new_line

fxch st5
fld qword [zero]
fcomp ; x1 ? 0
jb neg_root
je .one_root1

fsqrt
fchs
fstp [buffer]
mov rax, [buffer]
push rax
call print_float
```

```
    add rsp, 8
    call new_line
    call exit
    fchs

    .one_root1:
        fstp [buffer]
        mov rax, [buffer]
        push rax
        call print_float
        add rsp, 8
        call new_line

    call exit


zeroD:
    mov rsi, msg_zero_D
    call print_str
    call new_line

    fstp [buffer]
    mov rax, [buffer]
    push rax
    call print_float
    add rsp, 8
    call new_line

    fchs
    fstp [buffer]
    mov rax, [buffer]
```

```
    push rax
    call print_float
    add rsp, 8
    call new_line
    call exit


neg_D:
    mov rsi, msg_neg_D
    call print_str
    call new_line
    call exit

neg_root:
    call exit


;Function printing of string
;input rsi - place of memory of begin string
print_str:
    push rax
    push rdi
    push rdx
    push rcx
    mov rax, rsi
    call len_str
    mov rdx, rax
    mov rax, 1
    mov rdi, 1
    syscall
    pop rcx
```

```
    pop rdx
    pop rdi
    pop rax
    ret


len_str:
  push rdx
  mov rdx, rax
  .iter:
      cmp byte [rax], 0
      je .next
      inc rax
      jmp .iter
  .next:
     sub rax, rdx
     pop rdx
     ret


exit:
    mov rax, 1
    mov rbx, 0
    int 0x80
```

Анализ результатов

```
PS C:\Users\User\Projects\sem3\lab1_asm> ./lab1_asm
Два корня -1 1
```

```
PS C:\Users\User\Projects\sem3\lab1_asm> ./lab1_asm
Нет корней
```

```
PS C:\Users\User\Projects\sem3\lab1_asm> ./lab1_asm
Корни уровнения -4.952434787393863 -0.27090529085106513 0.27090529085106513 4.952434787393863
```