

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук Департамент программной инженерии  
Дисциплина: «Архитектура вычислительных систем»

**ПРОГРАММА, ВЫЧИСЛЯЮЩАЯ ЗНАЧЕНИЕ  
ЭКСПОНЕНТЫ ОТ ВВЕДЕННОГО ЧИСЛА**

Пояснительная записка

Выполнил:  
Кононов Алексей  
БПИ198

Москва  
2020

## Содержание

1. Текст задания .....	2
2. Применяемые расчетные методы .....	2
3. Описание взаимодействия с программой .....	2
4. Дополнительный функционал программы .....	2
5. Тестирование программы .....	3
5.1. Корректные значения.....	3
5.2. Некорректные значения.....	5
ПРИЛОЖЕНИЕ 1.....	8
Список литературы .....	8
ПРИЛОЖЕНИЕ 2.....	9
Код программы .....	9

## 1. Текст задания

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,1% значение функции  $e^x$  для заданного параметра  $x$  (использовать FPU).

## 2. Применяемые расчетные методы

Для решения задачи необходимо воспользоваться степенными рядами. Известно, что экспонента при разложении в ряд Маклорена имеет следующий вид:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Программа хранит в стеке информацию о следующем слагаемом, текущей сумме, номере слагаемого. На каждой итерации цикла сумма изменяется, результат подсчета становится «точнее». Слагаемое домножается на  $x$  и делится на номер итерации. Для проверки необходимости завершения цикла подсчета в стек вводится значение одной тысячной от текущей суммы и сравнивается с модулем текущего слагаемого.

## 3. Описание взаимодействия с программой

При запуске программы вводится строка, сообщающая о том, что пользователь может ввести вещественное число. На следующей строке пользователь вводит число. На следующей строке выводится строка с результатом вычисления экспоненты от введенного числа и пояснением о том, что это результат вычисления экспоненты.

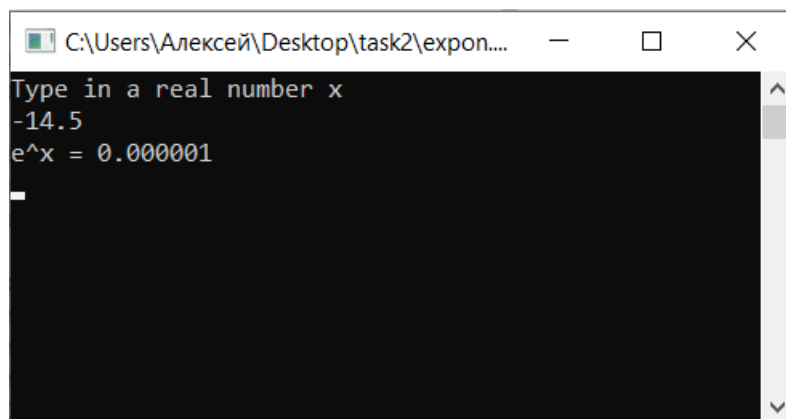
## 4. Дополнительный функционал программы

Закомментирована версия программы, не использующая степенной ряд для вычисления экспоненты. Описание работы программы написано в файле .asm.

## 5. Тестирование программы

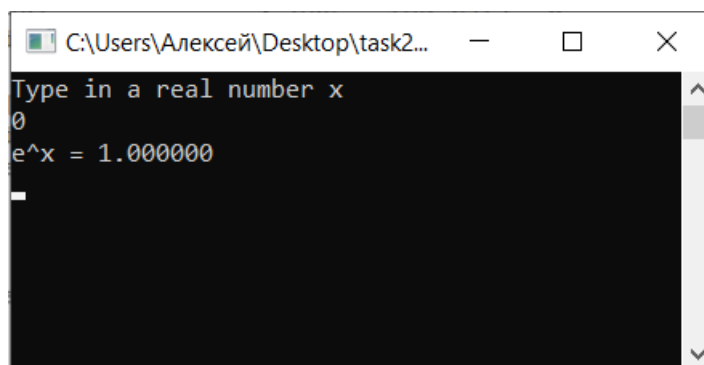
### 5.1. Корректные значения

Можно вводить числа не меньше -14,5 ввиду того, что выводится 6 знаков после запятой. Крайнее правое значение  $x = 709.79$  с точностью до 0,01.



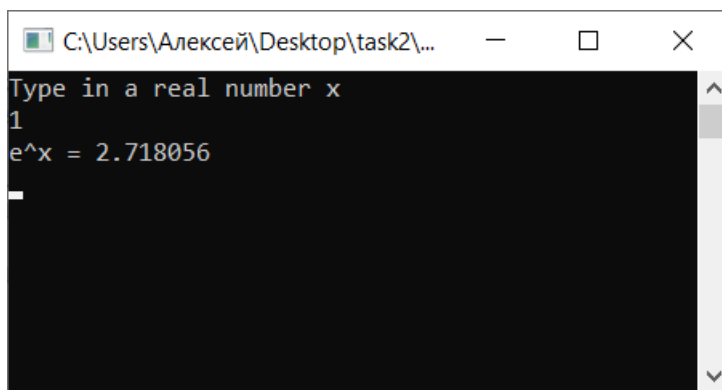
```
C:\Users\Алексей\Desktop\task2\expon....
Type in a real number x
-14.5
e^x = 0.000001
```

*Крайнее левое допустимое значение с точностью до 0,1*



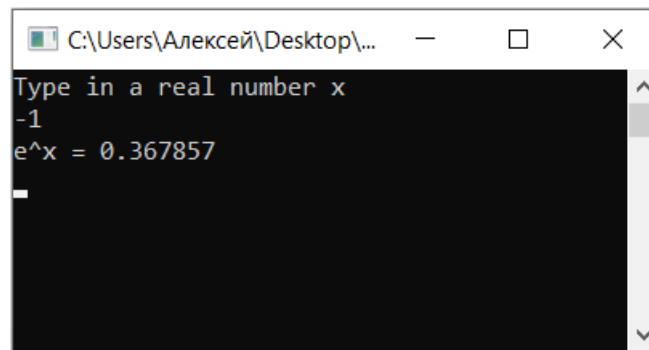
```
C:\Users\Алексей\Desktop\task2...
Type in a real number x
0
e^x = 1.000000
```

*Что происходит при введении 0 (то есть в цикл не зайдём)*



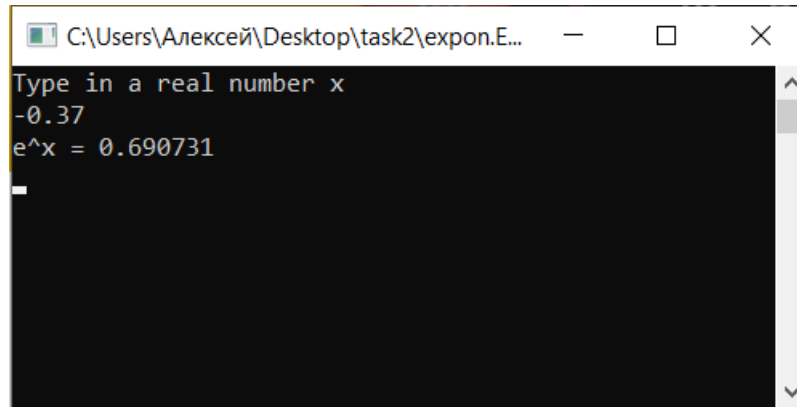
```
C:\Users\Алексей\Desktop\task2\...
Type in a real number x
1
e^x = 2.718056
```

*Ввод единицы*



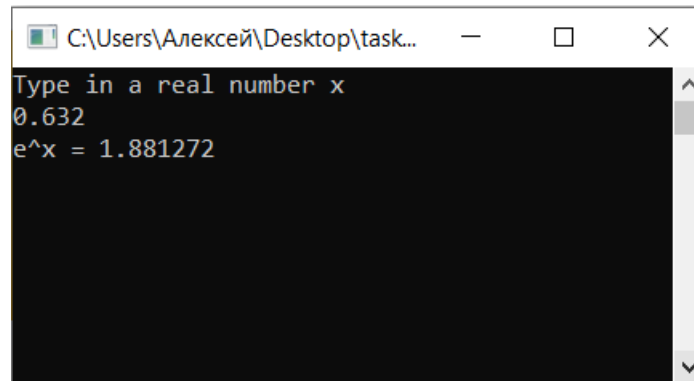
```
C:\Users\Алексей\Desktop\...  -  □  ×  
Type in a real number x  
-1  
e^x = 0.367857  
_
```

*Ввод минус единицы*



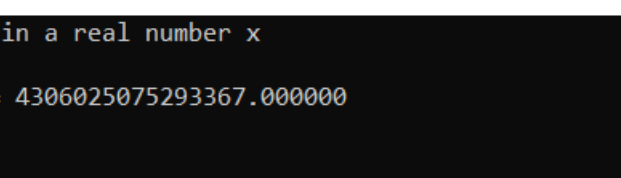
```
C:\Users\Алексей\Desktop\task2\expon.E...  -  □  ×  
Type in a real number x  
-0.37  
e^x = 0.690731  
_
```

*Ввод отрицательного по модулю меньше единицы*



```
C:\Users\Алексей\Desktop\task...  -  □  ×  
Type in a real number x  
0.632  
e^x = 1.881272  
_
```

*Ввод положительного по модулю меньше единицы*



C:\Users\Алексей\Desktop\task2\expon... — □ ×

Type in a real number x

36

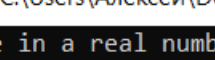
$e^x = 4306025075293367.000000$

### Ввод большого числа

[illegible]

Крайнее правое значение с точностью до 0.01 (ответ верный, для 709,8 выводится уже INF)

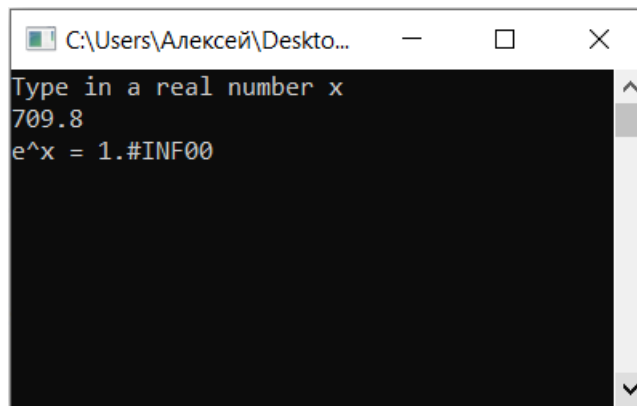
## 5.2. Некорректные значения



The screenshot shows a Windows command prompt window with the title bar "C:\Users\Алексей\Desktop...". The prompt is "Type in a real number x". The user has entered "-14.6". The output is "e^x = 0.000000".

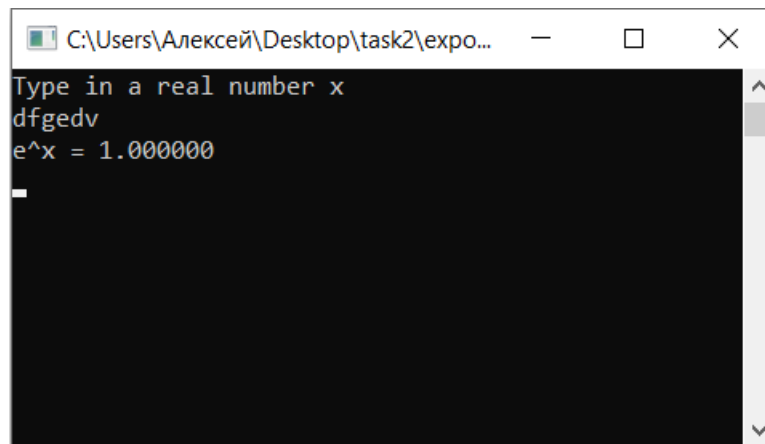
```
C:\Users\Алексей\Desktop...
Type in a real number x
-14.6
e^x = 0.000000
```

Первое значение за пределами допустимого диапазона



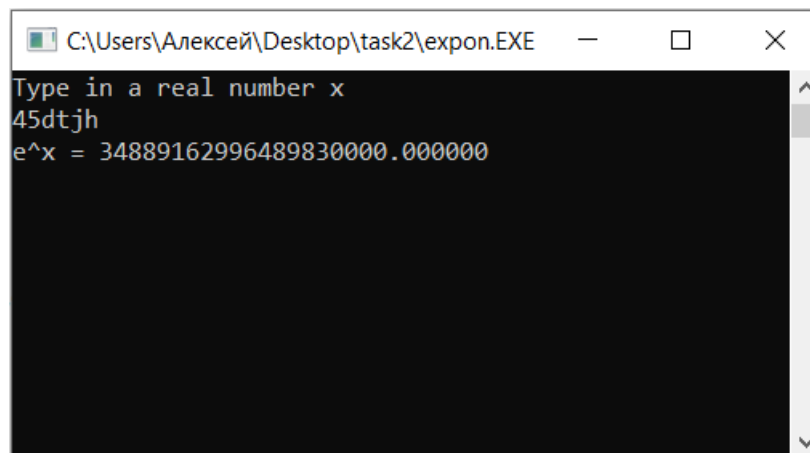
```
C:\Users\Алексей\Desktop...
Type in a real number x
709.8
e^x = 1. #INF00
```

*Второе значение за пределами диапазона*



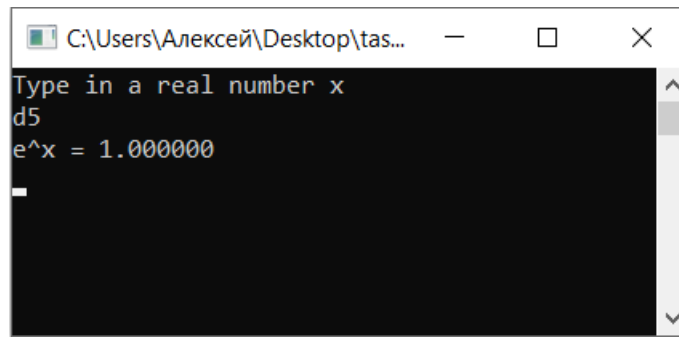
```
C:\Users\Алексей\Desktop\task2\expo...
Type in a real number x
dfgedv
e^x = 1.000000
```

*При вводе строки  $x = 0$*



```
C:\Users\Алексей\Desktop\task2\expon.EXE
Type in a real number x
45dtjh
e^x = 34889162996489830000.000000
```

*Число + строка распознается как число*



```
C:\Users\Алексей\Desktop\tas...
Type in a real number x
d5
e^x = 1.000000
_
```

*Строка + число не распознается,  $x = 0$*



**Список литературы**

1. Презентация к семинару «использование сопроцессора с плавающей точкой» [softcraft.ru] // <http://softcraft.ru/edu/comparch/practice/asm86/05-fpu/fpu.pdf> 28.10.2020
2. Учебник по ассемблеру для «квалифицированных чайников» <http://osinavi.ru/asm> 28.10.2020

## Код программы

```

format PE console
entry start
include 'win32a.inc'

section '.data' data readable writeable
ns dd ?
hout dd ?
c dq ?
e dq ?
x dq ?
sum dq 1.00
iter dq 0.00
zer dq 0.00
one dq 1.00
percentage dq 0.001
strs db 'e^x = %lf', 10, 0
strIn dq '%lf', 10, 0
frstStr db 'Type in a real number x ', 10, 0

section '.code' code readable executable
start:

    invoke printf, frstStr
    invoke scanf, strIn,x
    finit
    fld [sum] ; добавление значений суммы, единицы (слагаемого)
и
    fld [one] ; номера итерации в стек
    fld [iter]
    jmp sumCalc
cont:

getOut: ; завершение программы
    fstp st0 ; очистка стека от модуля слагаемого
    fstp st0 ; очистка стека от тысячной доли суммы
    fst [c]
    invoke printf, strs, dword[c], dword[c+4] ; вывод результата
    invoke getch
    invoke ExitProcess,0

sumCalc: ; цикл подсчета суммы
    fadd [one] ; iter++
    fxch st1 ; st0 = a_{i-1}
    fmul [x] ; a_i = a_{i-1} * x / iter

```

fdiv st0, st1

```

    fxch st2                ; st0 = sum
    fadd st0, st2           ; sum += a_i
    fld st0                 ; st0 = sum (все остальные ячейки стека сдвигаются)
    fmul [percentage]      ; st0 *= 0.001

    fld st3                 ; st0 = a_i (все остальные ячейки стека сдвигаются)
    fabs                    ; st0 = |st0|
    fxch st1                ; swap |a_i|, sum*0.001

    fcomi st1               ; сравнить |a_i|, sum*0.001
    ja getOut               ; если sum*0.001 больше, выход из программы

    fstp st0                ; очистка стека от модуля слагаемого
    fstp st0                ; очистка стека от тысячной доли суммы
    fxch st2                ; подготовка стека к следующей итерации цикла, то
бишь
    fxch st1                ; восстановление последовательности значений в
цикле
    jmp sumCalc             ; повтор цикла

```

jmp cont

section '.idata' import data readable writeable

```

library kernel32, 'kernel32.dll', \
user32, 'user32.dll', \
msvcrt, 'msvcrt.dll'

```

```

include '\api\kernel32.inc'
include '\api\user32.inc'

```

```

import msvcrt, \
    printf, 'printf', \
    sprintf, 'sprintf', \
    scanf, 'scanf', \
    getch, '_getch'

```

; Программа подсчета без использования степенных рядов

```

;format PE console
;entry start
;include 'win32a.inc'

```

```

;section '.data' data readable writeable
;ns dd ?
;hout dd ?

```

```

; c dq ?
; e dq ?
; x dq ?
; zer dq 0.00
; strs db '%lf', 10, 0
; strIn dq '%lf', 10, 0
; ifBig db 'st0 is bigger', 10, 0
; ifSmall db 'st0 is smaller', 10, 0
; fsc db '2 power int', 10, 0

; section '.code' code readable executable
; start:

                                ; идея в том, чтобы взять константу log_2(e) и воспользоваться командой
f2xm1                           ;  $e^x = e^{(\log_2(e) * x)}$ 
                                ; проблема в том, что f2xm1 работает для чисел из промежутка (-1,1)
                                ; нужно  $\log_2(e) * x$  разбить на целую и дробную часть, отдельно
                                ; вычислить
                                ;  $2^{\lfloor \log_2(e) * x \rfloor}$  и  $2^{(\log_2(e) * x - \lfloor \log_2(e) * x \rfloor)}$ 
                                ; и сложить
;   invoke scanf, strIn, x
;   finit
;   fld [x]
;   fldl2e
;   fmul st0, st1      ;  $st0 = (\log_2(e) * x)$ 
;   fst [c]
;   fst st1           ;  $st2 = \log_2(e) * x$ 
;   frndint           ;  $st0 = \lfloor \log_2(e) * x \rfloor$ 
;
;   fst [e]
;   fcomi st1
;   ja decrem
; cont:
;   fxch st1
;   fsub st0, st1
;   fst st2           ;  $st2 = \log_2(e) * x - \lfloor \log_2(e) * x \rfloor$ 
;   fstp st0 ; pop st0
;   fld1
;   fscale      ;  $st0 = 2^{st1} == 2^{\lfloor \log_2(e) * x \rfloor}$ 
;   fld st2
;   f2xm1
;   fld1
;   faddp st1, st
;   fmul st0, st1
;   fst [c]
;   invoke printf, strs, dword[c], dword[c+4]
; invoke getch
; invoke ExitProcess, 0
; decrem:

```

```
;fld1  
;fsubp st1, st0  
;jmp cont
```

```
;section '.idata' import data readable writeable
```

```
;library kernel32,'kernel32.dll',\  
;user32, 'user32.dll',\  
;msvcrt, 'msvcrt.dll'
```

```
;include "\api\kernel32.inc"  
;include "\api\user32.inc"
```

```
;import msvcrt,\  
;    printf, 'printf',\  
;    sprintf, 'sprintf',\  
;    scanf, 'scanf',\  
;    getch, '_getch'
```