

Пояснительная записка к микропроекту №1
**Программа, вычисляющая значение
функции гиперболического тангенса**

НИУ ВШЭ, ДПИ
Тарасюк Инна Валерьевна, БПИ192(1)
Вариант 24

1 Условие задания

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,1% значение функции гиперболического тангенса

$$th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

для заданного параметра x (использовать FPU)

2 Математическое обоснование

2.1 Гиперболический тангенс

Исходную формулу для гиперболического тангенса можно преобразовать к следующему виду:

$$th(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Степенной ряд для функции e^{2x} :

$$e^{2x} = \sum_{n=0}^{+\infty} \frac{(2x)^n}{n!}$$

2.2 Точность вычислений

Согласно условию, точность вычислений должна быть следующей:

$$\frac{|\text{сумма} - \text{предыдущая сумма}|}{|\text{предыдущая сумма}|} < 0.1\%$$

3 Работа программы

3.1 Переменные

1. x - параметр x , вводимый пользователем
2. `two` - константа служит для умножения введенного значения x на 2
3. `numerator` - числитель текущего элемента суммы ряда e^{2x}
4. `counter` - счётчик для вычисления знаменателя текущего элемента суммы ряда e^{2x}
5. `factorial` - знаменатель текущего элемента суммы ряда e^{2x}
6. `denominator` - знаменатель дроби $th(x)$
7. `sum` - сумма ряда до $n^{\text{ГО}}$ члена; также переменная служит для сохранения значения числителя дроби $th(x)$ в функции `Result`

8. prev - сумма ряда до $n - 1^{10}$ члена
9. acc - константа - относительная погрешность = 0.1%
10. tmp_esp - указатель на стек в функции Sum
11. borderAbove - константа = минимальный положительный x, при котором значение функции гиперболического тангенса равно 1
12. borderUnder - константа = максимальный отрицательный x, при котором значение функции гиперболического тангенса равно -1
13. fmt_full_output - форматированный вывод ответа
14. strX - строка для сообщения о вводе значения x

3.2 Алгоритм

Программе на вход подается число x. С помощью переменной two значение x удваивается. В функции NextSum вычисляется значение текущего элемента ряда e^{2x} , затем это значение записывается в переменную sum, хранящую общую сумму ряда. Процесс продолжается до тех пор, пока не достигнута необходимая точность. Далее в функции Result высчитывается итоговое значение $th(x)$ путем вычисления числителя, равного $e^{2x} - 1$, и знаменателя, равного $e^{2x} + 1$.

4 Список функций

4.1 InputX

Функция считывает введенный пользователем параметр x. Если данные введены некорректно (неверными данными считается произвольный набор символов, не содержащий цифры), функция повторно вызывается и пользователь получает сообщение о необходимости ввести число заново. Иначе введенное число сравнивается с минимальным положительным и максимальным отрицательным значением x, при котором значение функции гиперболического тангенса становится равным 1 и -1 соответственно. Если введенное число больше *borderAbove* или меньше *borderUnder*, результат выводится в консоль, и программа завершает работу. В противном случае запускается процесс вычисления $th(x)$ для введенного значения x.

4.2 Sum

Функция удваивает введенный параметр x, сохраняет предыдущую сумму ряда и вызывает *NextSum* до тех пор, пока точность не достигла заданного значения. Проверка точности осуществляется посредством вызова функции *CheckAccuracy*.

Результат выполнения функции - вычисленное с точностью *acc* значение функции e^{2x} для параметра *x*, записанное в *sum*.

4.3 NextSum

Функция вычисляет текущий элемент ряда e^{2x} , затем текущее значение суммы ряда. В процессе функции значение *counter* увеличивается на 1, значение *factorial* возрастает в *counter* раз, переменная *sum* увеличивается на $\frac{(2x)^n}{\text{factorial}}$

4.4 Result

Функция вычисляет значение гиперболического тангенса: высчитывается числитель, равный $e^{2x} - 1$, и знаменатель, равный $e^{2x} + 1$, дроби $th(x)$. Итоговое значение записывается в переменную *sum*.

4.5 CheckAccuracy

Функция проверяет, удовлетворяет ли вычисленное значение суммы заданной точности (описана в "Точность вычислений" 2.2). На основе сравнения устанавливает флаги. При удовлетворении условия на точность совершается переход по *JB*, в противном случае - по *JA*.

5 Формат выходных данных

5.1 Значение параметра x

На первой строке располагается информация о введенном пользователем значении *x*:

Input x = ***,
где * - введенные цифры

5.2 Значение гиперболического тангенса

Вычисленное значение гиперболического тангенса выводится на следующих строках в таком виде:

```
-----  
th(x) = *.*****  
-----  
, где * - конкретные цифры.
```

6 Текст программы

```
format PE console
entry start

include 'win32a.inc'

;th(x) = (e^(2x) -1) /(e^(2x)+1)

;-----first act - variables-----
section '.data' data readable writable

    fmt_full_output db "-----", 13, 10,\
                        "th(x) = %f", 13, 10,\
                        "-----", 13, 10, 0
    strX            db 'Input x: ', 0

    endBelow db "-----", 13, 10,\
                "th(x) = -1.000000",13,10,\
                "-----", 13, 10, 0
    endAbove db "-----", 13, 10,\
                "th(x) = 1.000000",13,10,\
                "-----", 13, 10, 0

    strScan db '%lf', 0
    sum dq 1.000 ; skip first element in sum(= 1)
    prev dq 0.000 ; sum on previous step
    acc dq 0.001 ; accuracy (0.1%)

    buf rd 100
    x dq 0
    numerator dq 1.000
    two dq 2.000
    denominator dq ?
    counter dd 0 ; loop counter
    factorial dq 1.000 ; factorial for current loop step
    borderAbove dq 3.800
    borderUnder dq -3.800
    tmp_esp dd ? ; used for saving stack ptr in Sum

;-----second act - script-----
section '.code' code readable executable

start:
```

```

    FINIT ; init FPU api

    call InputX

    call Sum ; count sum
    call Result
    invoke printf, fmt_full_output, dword[sum], dword[sum+4] ; print sum

    invoke getch
    invoke ExitProcess, 0 ; end program

;----- procedures -----
; Processing the entered value
InputX:

    push strX
    call [printf]
    add esp, 4

    invoke gets, buf
    invoke sscanf, buf, strScan, x
    add esp, 16

    cmp eax, 1
    jne InputX

    FLD [x]
    FCOMP [borderAbove]
    FSTSW ax ; FPU flags to ax
    WAIT

    sahf ; ax to CPU flags
    jb notAbove
        invoke printf, endAbove
        invoke getch
        invoke ExitProcess, 0

notAbove:
    FLD [x]
    FCOMP [borderUnder]
    FSTSW ax ; FPU flags to ax
    WAIT

    sahf ; ax to CPU flags

```

```

        ja notBelow
        invoke printf, endBelow
        invoke getch
        invoke ExitProcess, 0

notBelow:

        ret

; Count next sum, until it passes accuracy check
Sum:
    mov [tmp_esp], esp ; printf uses stack so we should save esp
    FLD [x]
    FMUL [two]
    FSTP [x]

acc_loop:
    FLD [sum]
    FSTP [prev] ; Save sum to prev via FPU stack
    WAIT

    call NextSum ; Count next sum approximation

    call CheckAccuracy ; Check if it fits approximation requirements
    ja acc_loop ; in case it doesn't fit - run loop again

    mov esp, [tmp_esp]
    ret

; count total th(x) value
Result:

    FLD1
    FADD [sum]
    FSTP[denominator] ; calculating the denominator of th(x)

    FLD1
    FSUBR[sum] ; calculating the numerator of th(x)

    FDIV[denominator]
    FSTP[sum] ;total th(x) value

    WAIT

```

```

    ret
; Count next sum approximation

NextSum:

    FLD[numerator]
    FMUL[x]          ; numerator *= x;
    FSTP[numerator]

    WAIT

    mov eax, [counter]
    inc eax
    mov [counter], eax ; eax = ++n

    FLD [factorial]
    FIMUL [counter]   ; factorial = counter!
    FSTP [factorial]

    FLD [numerator]
    FDIV [factorial]
    FADD [sum]
    FSTP [sum] ; sum = sum + (x / fact)

    WAIT

    ret

; Check, whether difference between sum and prev_sum is less then acc
CheckAccuracy:
    FLD [sum]
    FSUB [prev]
    FABS ; st(0) = abs(sum - prev)
    FDIV [sum] ; st(0) = abs(sum - prev) / sum

    FABS      ; abs((sum-prev)/ sum)
    FCOMP [acc]
    FSTSW ax ; FPU flags to ax
    WAIT

    sahf ; ax to CPU flags

```



```

ret

;-----third act - includes-----
section '.idata' import data readable
    library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll',\
        user32, 'USER32.DLL'

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

import kernel,\
    ExitProcess, 'ExitProcess',\
    HeapCreate, 'HeapCreate',\
    HeapAlloc, 'HeapAlloc'
include 'api\kernel32.inc'
import msvcrt,\
    printf, 'printf',\
    sprintf, 'sprintf',\
    sscanf, 'sscanf',\
    scanf, 'scanf',\
    puts, 'puts',\
    getchar, 'getchar',\
    fflush, 'fflush',\
    gets, 'gets',\
    getch, '_getch'

```

7 Тестирование программы

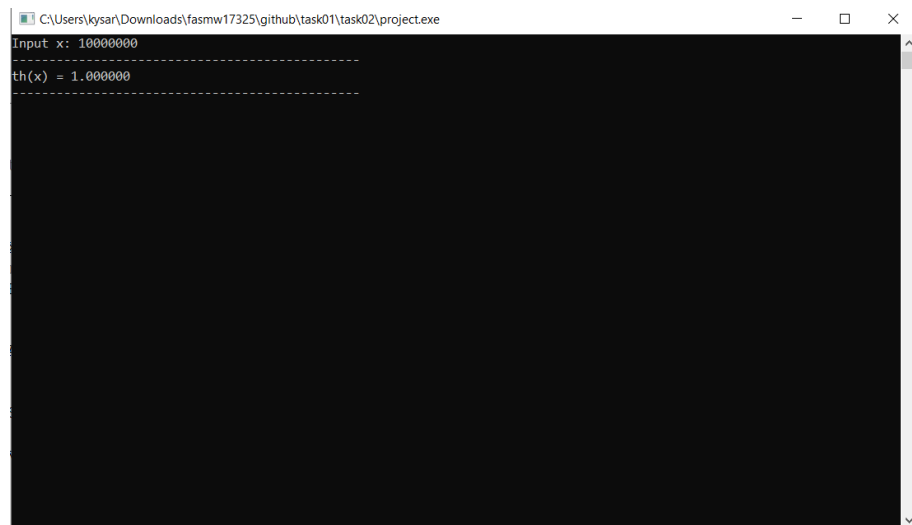
7.1 Корректные данные

При введении корректных данных программа работает исправно:

```
C:\Users\kysar\Downloads\fasmw17325\github\task01\task02\project.exe
Input x: 3.2
-----
th(x) = 0.996681
-----
```

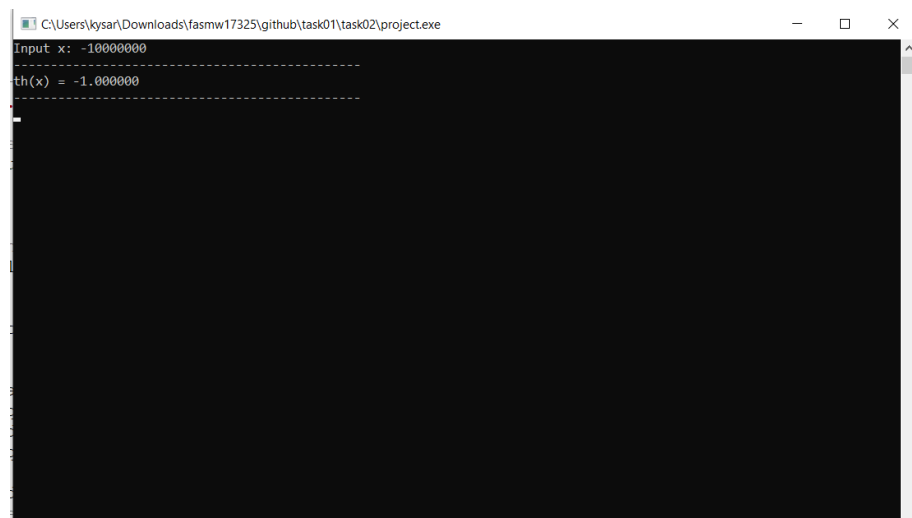
```
C:\Users\kysar\Downloads\fasmw17325\github\task01\task02\project.exe
Input x: 2.0
-----
th(x) = 0.964018
-----
add_aq 0.964018, add_aq 0.964018,
```

Правильную работу программы при введении большого положительного значения x можно увидеть на следующем изображении:



```
C:\Users\kysar\Downloads\fasmw17325\github\task01\task02\project.exe
Input x: 10000000
-----
th(x) = 1.000000
-----
```

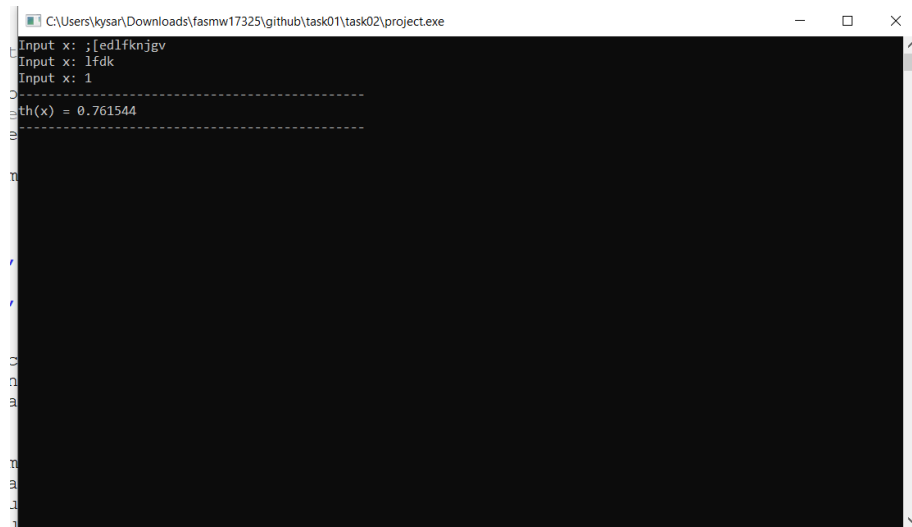
Тестирование программы при введении большого отрицательного значения x :



```
C:\Users\kysar\Downloads\fasmw17325\github\task01\task02\project.exe
Input x: -10000000
-----
th(x) = -1.000000
-----
```

7.2 Некорректные данные

Программа также предусматривает обработку неверных данных. При некорректном вводе программа просит пользователя повторно ввести параметр x до тех пор, пока введенное значение не будет верным. Результат работы программы можно увидеть на следующей фотографии:



```
C:\Users\kysar\Downloads\fasmw17325\github\task01\task02\project.exe
Input x: ;[edlfknjgv
Input x: Ifdk
Input x: 1
-----
th(x) = 0.761544
-----
```

8 Репозиторий исходного кода

github: <https://github.com/InnaTarasyuk/project1.asm/tree/main/project1.asm/project1>

9 Источники

1. информация о гиперболическом тангенсе https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D0%B1%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8
2. информация о fasm <http://flatassembler.narod.ru/fasm.htm>
3. информация о FPU <http://www.club155.ru/x86cmdfpu>
4. информация о FPU <https://prog-cpp.ru/asm-coprocessor-command/>
5. информация о FPU <http://osinavi.ru/asm/FPUexpansion/5.html>
6. <http://softcraft.ru/>
7. Wolfram для проверки значений $\text{th}(x)$: <https://www.wolframalpha.com/>