

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
РЕАЛИЗАЦИЯ ФУНКЦИИ ГИПЕРБОЛИЧЕСКОГО СИНУСА СРЕДСТВАМИ
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ FASM**

Автор пояснительной записки, _____
студентка ФКН, направление
“Программная инженерия”,
группа БПИ193

Н.А. Доскач

Москва 2020

1. ПОСТАНОВКА ЗАДАЧИ

В рамках данной работы я реализовала вычисление функции гиперболического синуса с точностью не хуже 0,1% с помощью степенного ряда средствами Flat Assembler [1], а также провела несколько тестов, проверяющих корректность работы алгоритма.

Точный текст задания: “Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,1% значение функции гиперболического синуса $sh(x) = \frac{e^x - e^{-x}}{2}$ для заданного параметра x (использовать FPU)”.

2. ТЕОРЕТИЧЕСКАЯ СПРАВКА О ВЫЧИСЛЕНИИ ГИПЕРБОЛИЧЕСКОГО СИНУСА С ПОМОЩЬЮ СТЕПЕННОГО РЯДА

Гиперболический синус имеет формулу: $sh(x) = \frac{e^x - e^{-x}}{2}$ [2]. Для вычисления функции с помощью степенного ряда применяется следующая формула:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} * x^n \quad [2]$$

$$\begin{aligned} \text{Таким образом, } sh(x) &= \frac{e^x - e^{-x}}{2} = \frac{\left(\frac{e^x - e^{-x}}{2}\right)'}{1!} x + \frac{\left(\frac{e^x - e^{-x}}{2}\right)''}{2!} x^2 + \frac{\left(\frac{e^x - e^{-x}}{2}\right)'''}{3!} x^3 + \dots = \\ &= x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2n+1}}{(2n+1)!} + \dots = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}. \end{aligned}$$

3. ОПИСАНИЕ ОБЛАСТИ ДОПУСТИМЫХ ЗНАЧЕНИЙ ВХОДНЫХ ПАРАМЕТРОВ

Используются вещественные числа двойной точности (double), имеющие максимальные значения $[1.7 * 10^{308}, 1.7 * 10^{308}]$. Так как значение $sh(x)$ при $|x| > 710.0$ выходит за эти рамки, допустимые входные значения: $[-710.0, 710.0]$. При попытке ввода значений за этими границами пользователь получает сообщение об ошибке (“x must be between -710.0 and 710.0”) и программа завершается.

4. КОД ПРОГРАММЫ

```
format PE console
entry start

include 'win32a.inc'

section '.code' code readable executable

start:
    ; input x
    invoke printf, mesInput
    invoke scanf, formatF, x;

    FINIT
    ;first element of the sum is x, so sum after 1st step is x
    fld [x]
    fst [elem]
    fst [sum]
    ;check if x==0
    ftst
    fstsw ax
    sahf
    jz zero
    ;check x boundaries (must be between -710,710,otherwise print error message)
    fabs
    fstp [tmp]
    fld [boundary]
    fsub [tmp]
    ftst
    fstsw ax
    sahf
    jb tooBig ;print error message
;calculate loop
addElem:
    fld [sum]
    fstp [prevsum] ;prevSum = sum
    ; calculate element = element * x * x / ((2 * n) * (2 * n + 1)):
    ;1) calculate numerator (element * x * x)
    fld [elem]
    fmul [x]
    fmul [x]
    ;2)calculate denominator ((2 * n) * (2 * n + 1))
    mov eax,[n] ; eax = n
    mov ebx,2 ;ebx = 2
    mul ebx ;eax = 2n
    mov [denominator],eax ; denominator = 2n
    mov ebx,1 ;ebx = 1
    add eax,ebx ;eax = 2n +1
    mul [denominator] ; ; eax = 2n * (2n + 1)
    mov [denominator],eax
    ;3)calculate element = numerator / denominator = element * x * x / ((2 *
n) * (2 * n + 1))
    fidiv [denominator]
    fstp [elem]
    ;sum += element;
    fld [sum]
    fadd [elem]
    fstp [sum]
    ; n++;
    mov eax,[n]
    add eax, 1
    mov [n], eax
```

```

; if |(sum - prevSum) / sum| < 0.001 => end calculation
fld [sum]
fsub [prevsum] ; sto = sum - prevSum
fddiv [sum] ;st0 = (sum - prevSum / sum)
fabs
fstp [tmp]
fld [error]
fsub [tmp] ;st0 = 0.001 - (sum - prevSum / sum)
ftst
fstsw ax
sahf

jb contCalc ;need to continue the calculation
jmp printRes

tooBig: ;x is too big - print error message
invoke printf, mesError
jmp end_work

contCalc: ;clear stack and loop one more time
fstp [tmp]
jmp addElem

zero: ;sh(0)=0
invoke printf, mesRes, dword[x], dword[x+4], \
        dword[x], dword[x+4]
jmp end_work

printRes:

invoke printf, mesRes, dword[x], dword[x+4], \
        dword[sum], dword[sum+4]

jmp end_work

end_work:
invoke getch
invoke ExitProcess, 0

;-----third act - including HeapApi-----
-----
section '.data' data readable writable

mesInput      db      "To calculate sh(x) input x: ", 0
formatF       db      '%lf', 0
mesRes        db      "sh(%lf)= %lf", 10, 0
mesError      db      "x must be between -710.0 and 710.0", 10, 0
mesRealResult db      "x must be between -710.0 and 710.0", 10, 0
x dq         ? ;input
elem dq      0 ;element in a Taylor series
sum dq       0 ;sum of the sequence
prevsum dq   0 ;prevSum - sum before adding current element
tmp dq       ?
boundary dq   710.0 ;x must be between [-710, 710]
denominator dd 0 ;denominator of the element in taylor series (calculated
for each element)
error dq     0.001 ;than error < 0.1% - > end calculation
n dd 1 ;number of elements in Taylor series

section '.idata' import data readable
library kernel, 'kernel32.dll', \
        msvcrt, 'msvcrt.dll'

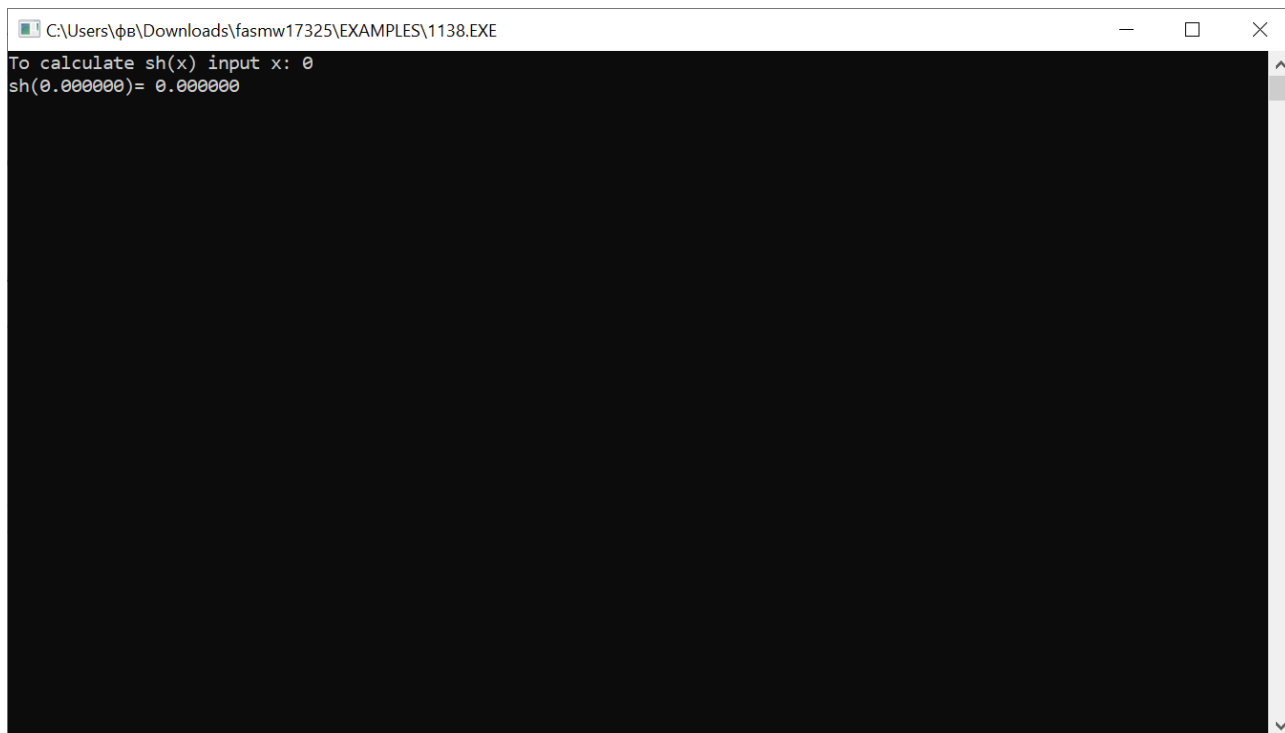
import kernel, \
        ExitProcess, 'ExitProcess'

```

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

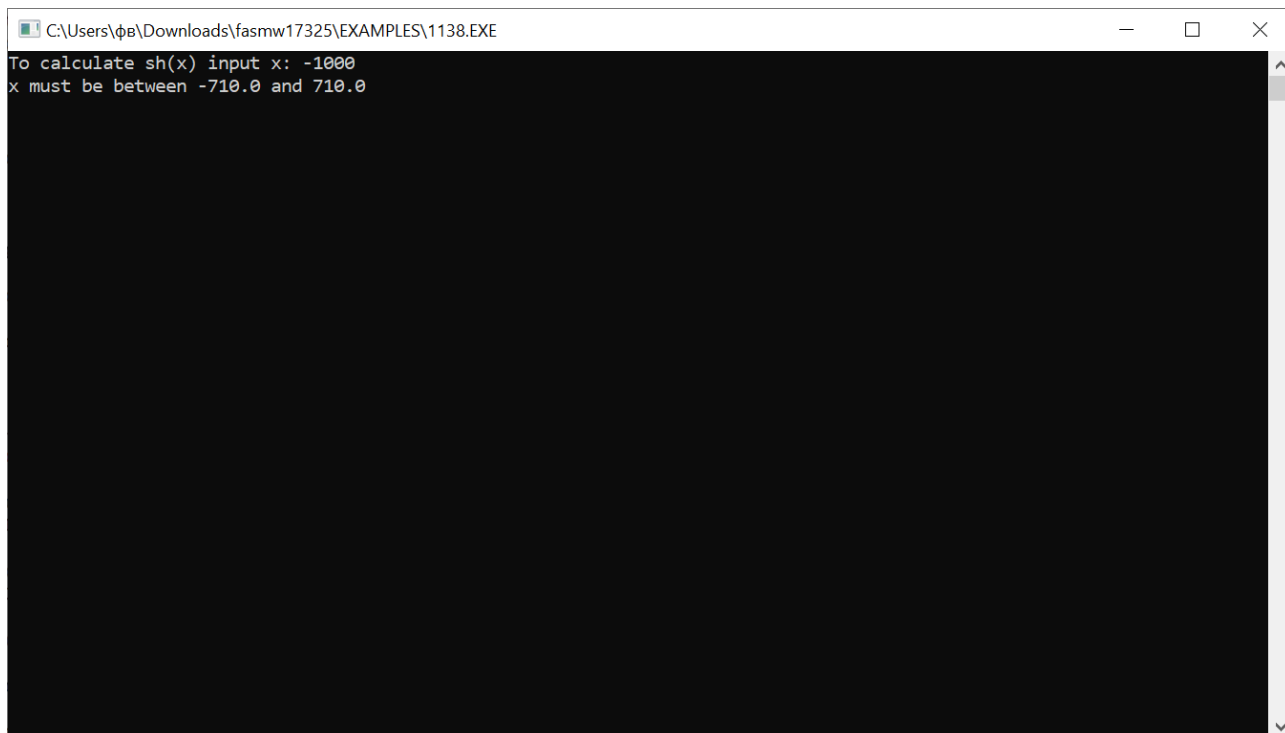
5. ТЕСТОВЫЕ ПРИМЕРЫ

Пример 1



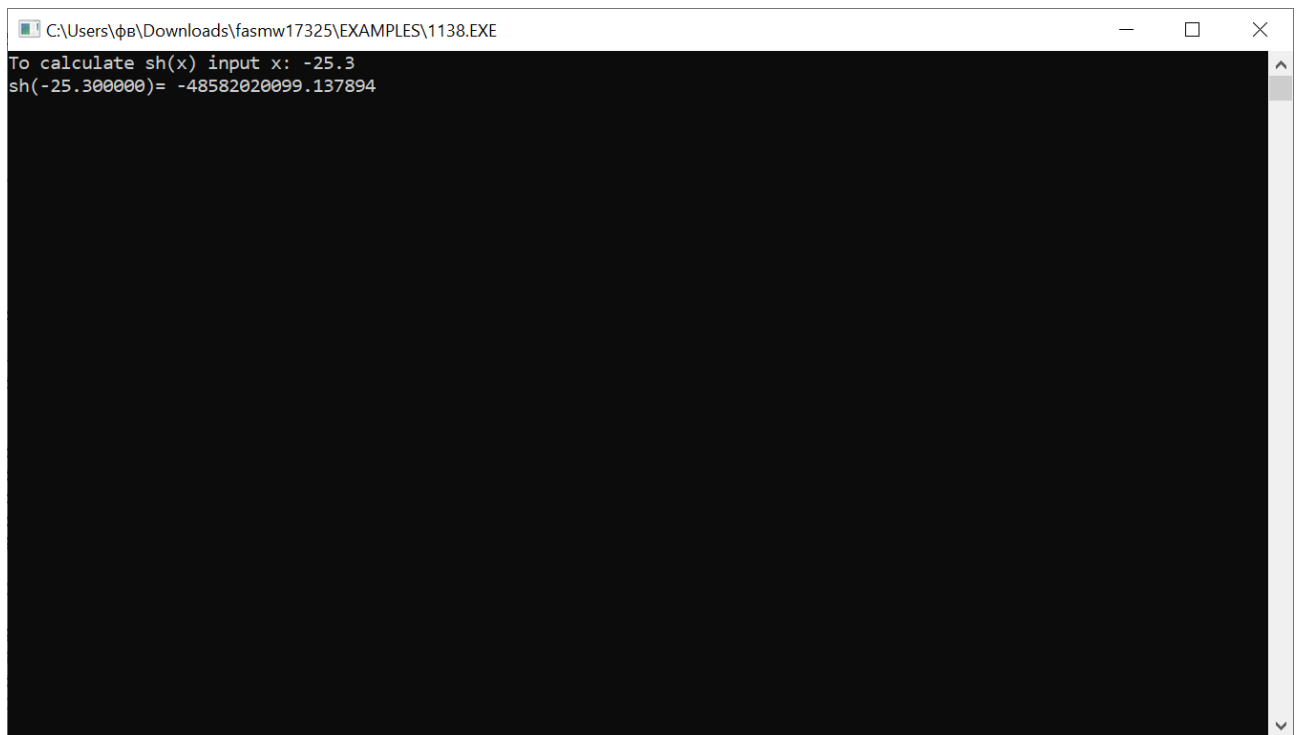
```
C:\Users\фв\Downloads\fasmw17325\EXAMPLES\1138.EXE
To calculate sh(x) input x: 0
sh(0.000000)= 0.000000
```

Пример 2



```
C:\Users\фв\Downloads\fasmw17325\EXAMPLES\1138.EXE
To calculate sh(x) input x: -1000
x must be between -710.0 and 710.0
```

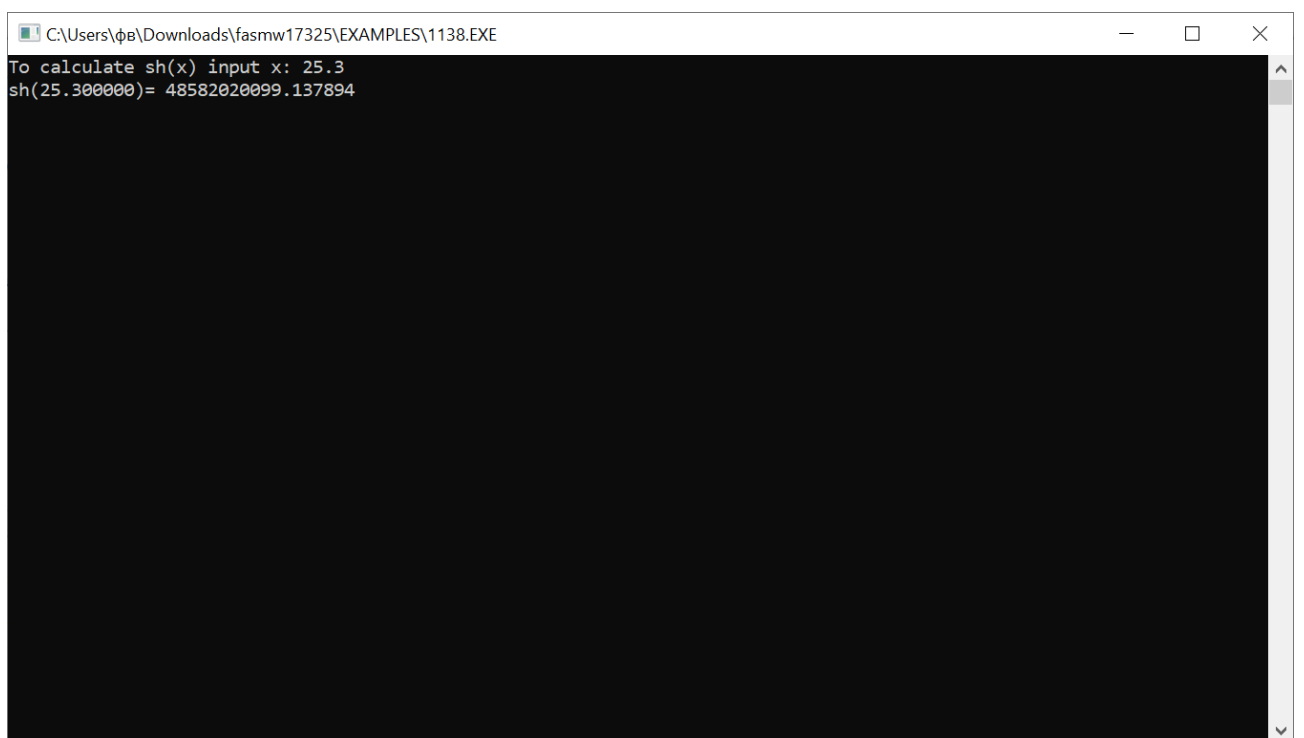
Пример 3



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\фв\Downloads\fasmw17325\EXAMPLES\1138.EXE. The window contains the following text:

```
To calculate sh(x) input x: -25.3  
sh(-25.300000)= -48582020099.137894
```

Пример 4



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\фв\Downloads\fasmw17325\EXAMPLES\1138.EXE. The window contains the following text:

```
To calculate sh(x) input x: 25.3  
sh(25.300000)= 48582020099.137894
```

Пример 5

6. СПИСОК ИСТОЧНИКОВ

1. Flat assembler. Documentation. [Электронный ресурс]. Режим доступа: <https://flatassembler.net/docs.php>, свободный. (дата обращения: 31.10.2020)
2. Григорий Фихтенгольц: Основы математического анализа. Учебник. Том 2. – Москва: Лань, 2018. - С. 462.
3. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления. – Москва: Издательство стандартов, 2001.