**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

**Пояснительная записка**

Исполнитель:  
студент группы **БПИ199**

**Шарипов Сардор Уткирович**

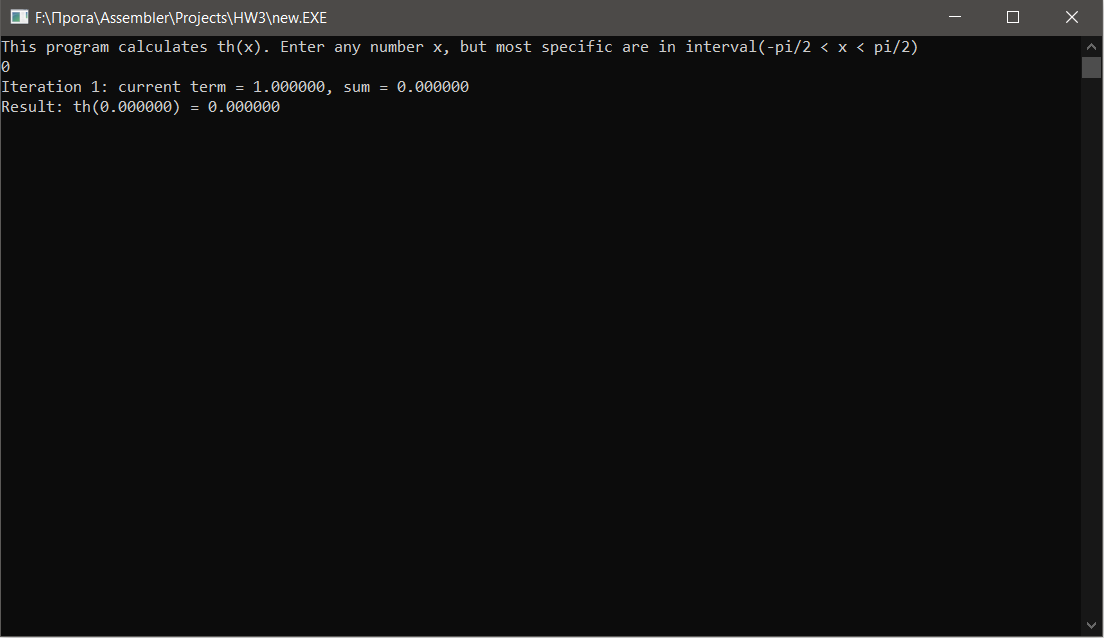
**Москва 2020**

1. **Текст задания**

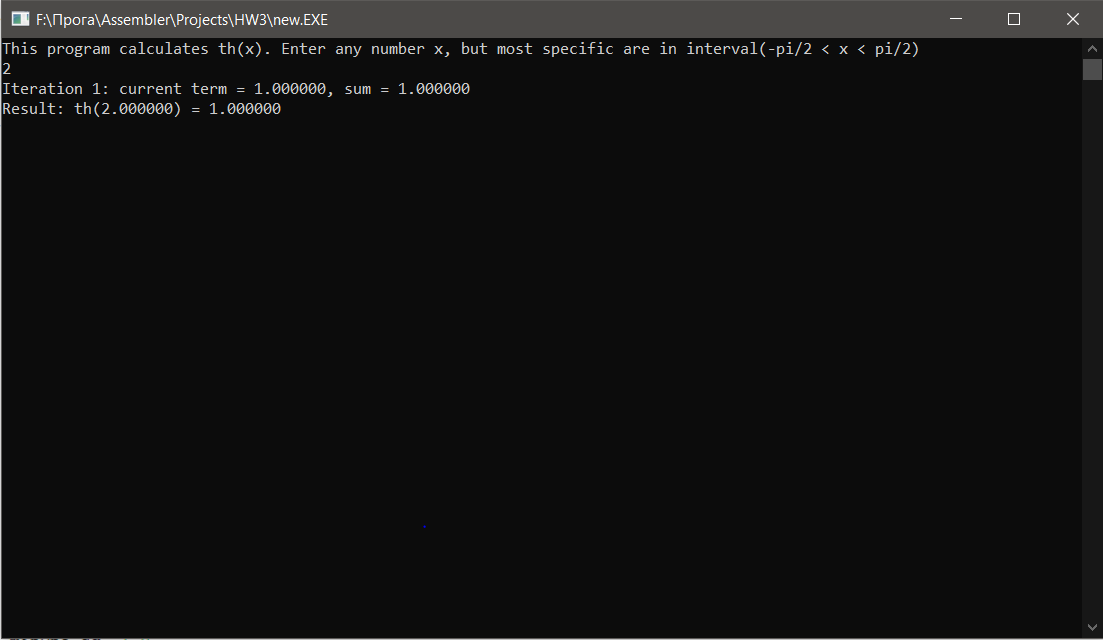
Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,05% значение функции гиперболического тангенса для заданного параметра x с точностью 0.05 % (использовать FPU)

1. **Описание алгоритма**
2. Пользователь вводит число , проверяется на корректность.
3. Вычисляем
4. ς(2k) где ς(2k) =
5. Проверяется на точность.
6. Выводится результат.
7. **Испытания**
   1. **Особые точки**

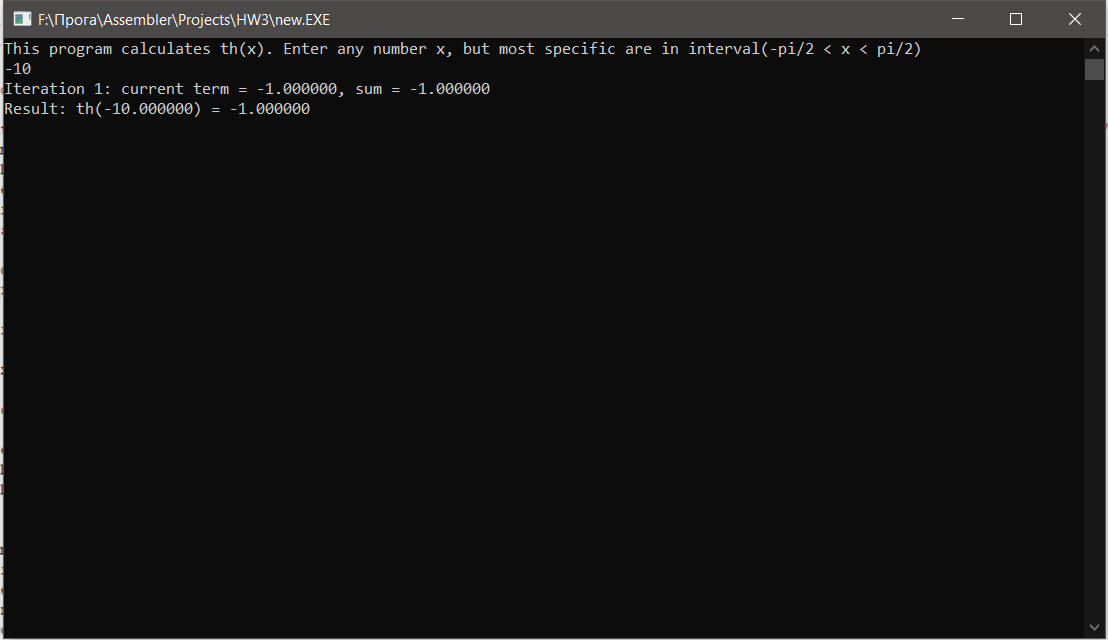
Х = 0, на калькуляторе – 0, программа:



X >= π/2, на калькуляторе – 1, программа:

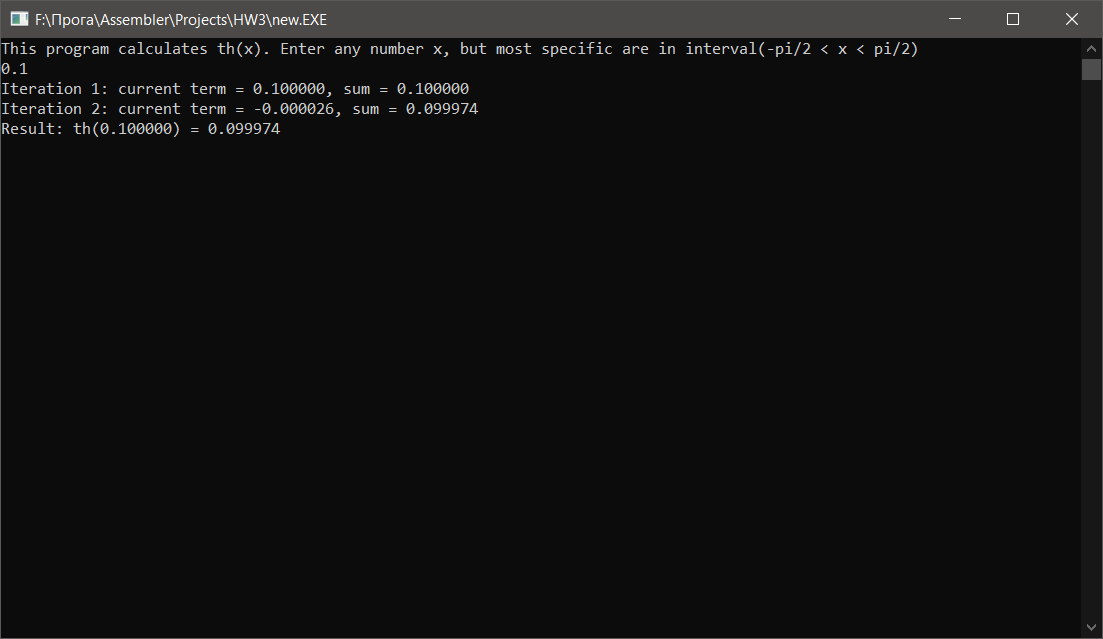


X <= π/2, на калькуляторе – (-1), программа:

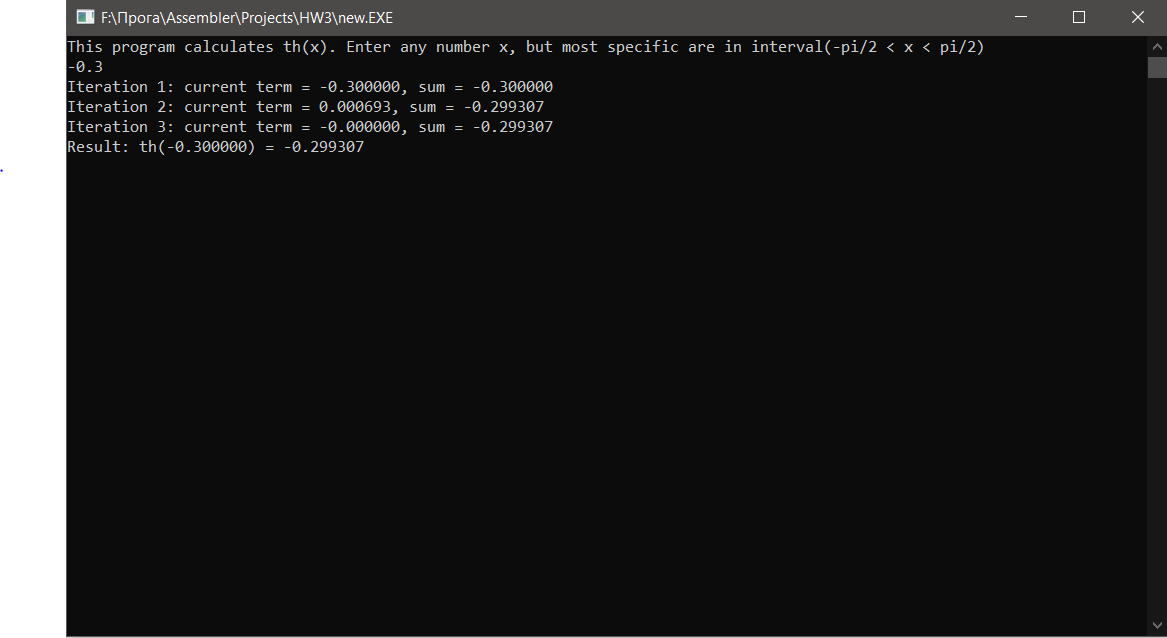


* 1. **На промежутке -π/2 <х < π/2**

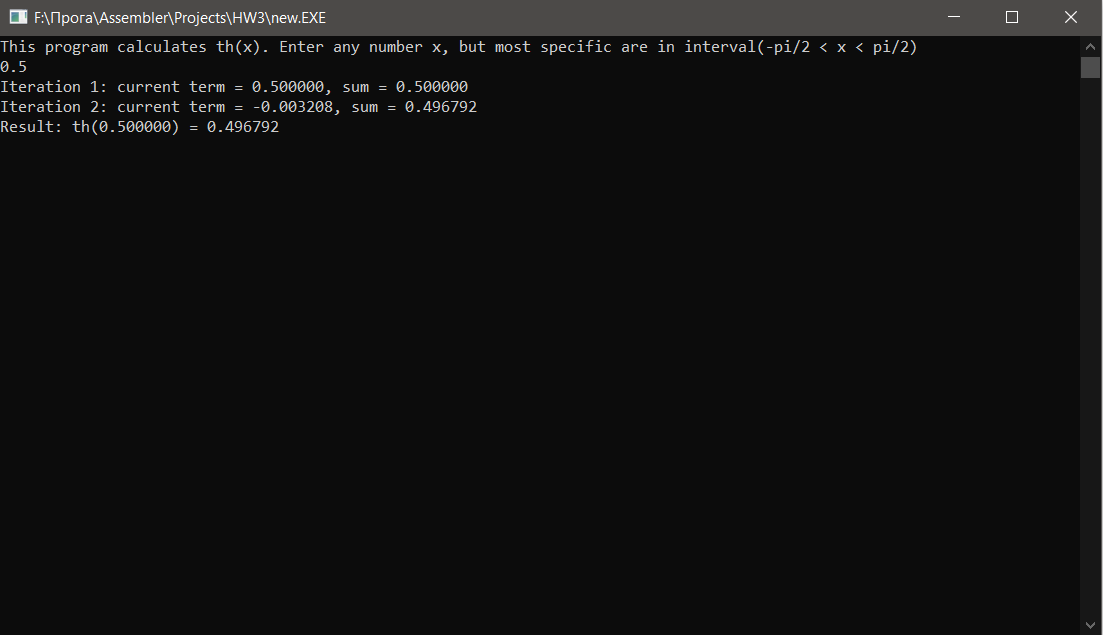
Х = 0.1, на калькуляторе - 0.099668, программа:



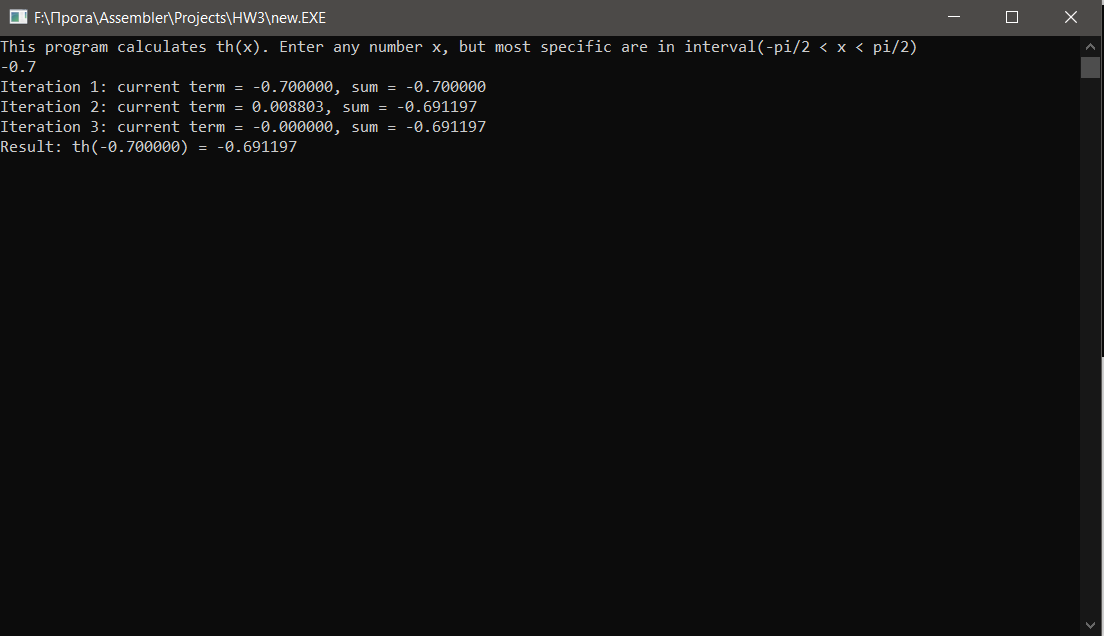
Х = -0.3, на калькуляторе – (-0.291313), программа:



Х = 0.5, на калькуляторе - 0.493117, программа:



Х = -0.7, на калькуляторе – (-0.684368), программа:



1. **Источники**

[https://en.wikipedia.org/wiki/Bernoulli\_number](https://en.wikipedia.org/wiki/Bernoulli_number%20) Числа Бернулли

<https://en.wikipedia.org/wiki/Riemann_zeta_function#The_functional_equation>Зета функция Римана

1. **Текст программы**
   1. **MacroHW.INC**
2. ; Макрос для вывода чисел с плавующей запятой.
3. macro PrintFloat string, [args] {
4. reverse
5. push dword[args + 4]
6. push dword[args]
7. common
8. push string
9. call [printf]
10. }
11. ; Макрос, инкрементирующий значение переменной word.
12. macro Inc value {
13. mov cx, [value]
14. inc cx
15. mov [value], cx
16. }
17. ; Макрос, инкрементирующий значение переменной dword.
18. macro Inc2 value {
19. mov ecx, [value]
20. inc ecx
21. mov [value], ecx
22. }
23. ; Макрос, копирующий значение одного числа с плавующей запятой в другое.
24. macro CopyFloat to, from {
25. mov eax, dword[from]
26. mov ebx, dword[from + 4]
27. mov dword[to], eax
28. mov dword[to + 4], ebx
29. }
30. ; Макрос для подсчета чисел Бернулли B\_2n
31. ; B\_2n  = (-1)^(n+1) \* 2/(2pi)^2n \* E(2n)
32. ; E(2n) = зета функция Эйлера
33. ; E(2k) = Sum (1/n)^2k
34. ; зедсь в коде  n как 2n в формуле
35. macro bernuolli result,n, one,minOne, two, inRes, inSum , pi, temp{
37. mov cx, [n]
38. fld [two]
39. b:
40. fdiv [two]
41. loop b ; 2/2^2n
42. mov cx, [n]
43. c:
44. fdiv [pi]
45. loop c ; 2/(2pi)^2n
46. fstp [result]
47. mov cx, [n]
48. d:
49. mov bx, cx
51. fld [one]
52. mov cx, [n]
53. e:
54. fdiv [temp]
55. loop e
56. fstp [inSum] ; 1/n^2k
57. fld [temp]
58. fsub [one]
59. fstp [temp]
60. fld [inSum]
61. fadd [inRes]
62. fstp [inSum] ; Сумма этих чисел
63. mov cx, bx
64. loop d
65. fld [result]
66. fmul [inSum]
67. fstp [result]
68. ; Смотря на n  берем знак числа бернулли
69. mov ax, [n]
70. fild [n]
71. fidiv word[two]
72. fiadd word[one]
73. fistp [n]
74. cmp ax, [n]
75. jpo odd
76. mov [n], ax
77. odd:
78. fld [result]
79. fchs
80. fstp [result]
81. ; это я темп на место возвращаю
82. mov cx, [n]
83. fld [temp]
84. f:
85. fadd [one]
86. loop f
87. fstp [temp]
88. }

**5.2. Main Program**

format PE Console

entry start

include 'F:\Прога\Assembler\INCLUDE\WIN32A.INC'

include 'F:\Прога\Assembler\Projects\HW3\MacroHW.INC'

section '.data' data readable writable

        enterStr db 'This program calculates th(x). Enter any number x, but most specific are in interval(-pi/2 < x < pi/2)', 10, 0

        numberStr db '%lf', 0

        debugStr db '%lf', 10, 0

        iterationStr db 'Iteration %d: ', 0

        currentValuesStr db 'current term = %lf, sum = %lf', 10, 0

        resultStr db 'Result: th(%lf) = %lf', 10, 0

        x dq ?  ;вводимое число

        sqrX dq ? ; квадрат х

        term dq 1.0 ; текущий член ряда

        nextTerm dq ? ; последующий член ряда

        factorialCounter dw 4

        iterationCounter dd 1

        subpow dd 2 ; 2^2n

        subpow2 dd 1 ; 2^2n-1

        sum dq 0.0 ;сумма ряда

        berRes dq ?; число Бернулли

        one dq 1.0

        minOne dq -1.0

        two dq 2.0

        inRes dq 0.0 ; промежуточные

        inSum dq 0.0 ;              переменные

        pi dq 3.141592

        temp dq 4.0

        epsilon dq 0.0005

section '.code' code readable executable

       start:

                FINIT

                call  inputNumber

                call  mainLoop

                call  outputResult

       finish:

                push 0

                invoke getch

                call ExitProcess

        zeroF:

                fld [term]

                fstp [nextTerm]

                call outputResult

                call finish

                ret

        minPiF:

                fld [one]

                fchs

                fst [nextTerm]

                fstp [sum]

                call outputResult

                call finish

                ret

        plusPiF:

                fld [one]

                fst [nextTerm]

                fstp [sum]

                call outputResult

                call finish

                ret

        inputNumber:

                invoke printf, enterStr

                invoke scanf, numberStr, x

                fld [sum]

                fld [x]

                ; производим сравнения

                fcompp

                fstsw ax

                sahf

                jz zeroF ;если 0 то функция равен 0

                fld [pi]

                fdiv [two]

                fld [x]

                fcompp

                fstsw ax

                sahf

                jae plusPiF ;если больше pi/2 то 1

                clc

                fld [pi]

                fdiv [two]

                fchs

                fld [x]

                fcompp

                fstsw ax

                sahf

                jbe minPiF ;если меньше pi/2 то -1

                fld [x]

                fmul [x]

                fstp [sqrX]; находим х^2

                fld [term]

                fmul [x]

                fstp [term]; первый член ряда равен х

                fld [term]

                fstp [sum]; сумма тоже х

        mainLoop:

                invoke printf, iterationStr,[iterationCounter]

                PrintFloat currentValuesStr, term, sum

                Inc2 iterationCounter

                fld [subpow]

                fimul [subpow]

                fstp [subpow]; получим квадрат (2^2n)в каждой итерации

                fld [subpow]

                fstp [subpow2]; получим (2^2n-1)

                Inc2 subpow2

                fld [term]

                fmul [sqrX]

                fstp [nextTerm] ; получим 2^2n-1 в каждой итерации

                bernuolli berRes,factorialCounter, one,minOne,two,inRes, inSum,pi,temp

                fld [nextTerm]

                fmul [berRes]   ;   умножаем

                fimul [subpow]   ;          полученные

                fimul [subpow2]   ;                    члены

                fstp [nextTerm]    ;

                fld [temp]

                fadd [two]      ; это как factorial iterator только qword

                fstp [temp]

                fld [sum]

                fadd [nextTerm]  ;суммируем

                fstp [sum]

                fld [sum]

                fabs

                fmul [epsilon]  ; сравниваем с эпсилон

                fld [nextTerm]

                fcompp

                fstsw ax

                sahf

                jb outputResult

                Inc factorialCounter

                Inc factorialCounter

                CopyFloat term, nextTerm

                jmp mainLoop

        ret

        outputResult:

                invoke printf, iterationStr,[iterationCounter] ;результат программы

                PrintFloat currentValuesStr, nextTerm, sum

                PrintFloat resultStr, x, sum

                call finish

        ret

section '.idata' import data readable

        library kernel, 'kernel32.dll',\

                msvcrt, 'msvcrt.dll'

        import kernel,\

               ExitProcess, 'ExitProcess',\

               GetProcessHeap, 'GetProcessHeap',\

               HeapAlloc, 'HeapAlloc'

        import msvcrt,\

               printf, 'printf',\

               scanf, 'scanf',\

               getch, '\_getch'