

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

**Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»**

**Пояснительная записка к первому микропроекту по дисциплине
“Архитектура вычислительных систем”**

Преподаватель: Легалов Александр Иванович

Студент: Мкртумян Роберт Манвелович

Группа: ФКН БПИ194

Москва 2020

Текст задания

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,05% значение функции гиперболического косинуса $\text{ch}(x) = (e^x + e^{-x})/2$ для заданного параметра x (использовать FPU)

Применяемые расчётные методы

Функцию гиперболического косинуса можно разложить на следующий степенной ряд:

$$\text{ch } x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}$$

Принцип решения:

Заведём переменную погрешности равную 0.0005(0.05% от 1)

Прибавляем к итоговому результату значение очередного члена степенного ряда до того момента пока это значение не меньше погрешности

Псевдокод решения представлен ниже на рисунке 1

```
begin
eps = 0.0005 ; погрешность
n = 0 ; индекс члена степенного ряда
t = 1 ; значение члена степенного ряда
s = t ; итоговое значение
while abs(t) >= eps do ; пока модуль очередного члена больше погрешности
begin
n:=n+1 ; индекс следующего члена ряда
t:=t*x*x/(2*n)/(2*n-1) ; домножаем предыдущий на x^2, делим на 2 очередных числа,
; получаем x^2n/(2n)!
s:=s+t ; прибавляем значение очередного члена к итоговому результату
end;
end;
```

Рисунок 1

Исходный текст программы

format PE console

entry start

include 'win32a.inc'

; _____ Data reserve section _____

section '.data' data readable writable

formatStr db '%s', 0

formatNum db '%lf', 0

const_2 dq 2.0 ; константа со значением 2

cosnt_1 dq 1.0 ; константа со значением 1

divider1 dq ? ; первый делитель текущего члена ряда

divider2 dq ? ; второй делитель текущего члена ряда

limit dq 101.0 ; максимальный по модулю аргумент(не включая)

n dq 0 ; индекс члена ряда

t dq 1 ; значение текущего члена ряда

s dq 0 ; итоговое значение

X dq ? ; значение, вводимое пользователем

eps dq 0.0005 ; epsilon(точность 0.05%)

strInputX db 'Enter the value of the function ch(x) argument in the range
[-100;100] - ', 0

strAnswer db 'ch(%lf) = %lf', 0

strIncorInput db 'the number %lf is not in the diaozone [-100;100]', 0

; _____ Code section _____

section '.code' code readable executable

start:

push strInputX ; Просим пользователя ввести аргумент функции

call [printf] ;

```

push X          ;

push formatNum  ; Записываем введенное значение в переменную X

call [scanf]    ;

fild [X]        ;

fabs            ; Сравниваем модуль введенного значения

fcomp qword [limit] ; С максимально возможным аргументом

fstsw ax        ;

sahf            ;

jnb stop        ; Если введенное значение вне диапазона,
завершаем программу

fild [s]        ;

fadd [cosnt_1]  ; Присваиваем первому члену ряда и итоговому результату

fst [s]         ; итоговому результату значение 1

fstp [t]        ;

calculateRow:

    fild [n]    ;

    fadd [cosnt_1] ; инкрементируем индекс члена ряда

    fstp [n]    ;

    fild [t]    ;

    fmul [X]    ; умножаем член ряда на X^2

    fmul [X]    ;

    fstp [t]    ;

```

```
fld [n] ;  
fmul [const_2] ; получаем первый делитель члена ряда(2n)  
fstp [divider1] ; записываем в divider1
```

```
fld [n] ;  
fmul [const_2] ; получаем второй делитель члена ряда(2n-1)  
fsub [const_1] ;  
fstp [divider2] ; записываем в divider2
```

```
fld [t] ;  
fdiv [divider1] ; делим член ряда на divider1(2n)  
fdiv [divider2] ; и на divider2(2*n-1)  
fstp [t] ;
```

```
fld [s] ; Прибавляем к итоговому результату  
fadd [t] ; значение члена ряда  
fstp [s] ;
```

```
fld [t] ;  
fabs ; сравниваем член ряда по модулю  
fcomp qword [eps] ; с погрешностью, если член ряда  
fstsw ax ; больше погрешности продолжаем цикл  
sahf ; иначе формируем результат и  
завершаем программу  
jnb calculateRow ;
```

```
finish: ; Завершаем программу и выводим результат  
invoke printf, strAnswer, dword[X], dword[X+4], dword[s], dword[s+4]
```

```
call [getch]

push 0

call [ExitProcess]
```

stop: ; Завершаем программу по причине некорректных входных данных

```
invoke printf, strIncorInput, dword[X], dword[X+4]

call [getch]

push 0

call [ExitProcess]
```

;_____Data import section_____

section '.idata' import data readable

```
library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll'
```

```
import kernel,\

ExitProcess, 'ExitProcess'
```

```
import msvcrt,\

printf, 'printf',\

getch, '_getch',\

scanf, 'scanf'
```

Тестирование

Функция $ch(x)$ определена на всей вещественной прямой, но, так как эта функция достаточно быстрорастущая, то при большом по модулю аргументе время работы программы значительно

повышается, поэтому в программе введено ограничение диапазона аргумента [-100;100]

Обработка некорректного ввода (рисунки 2, 3)

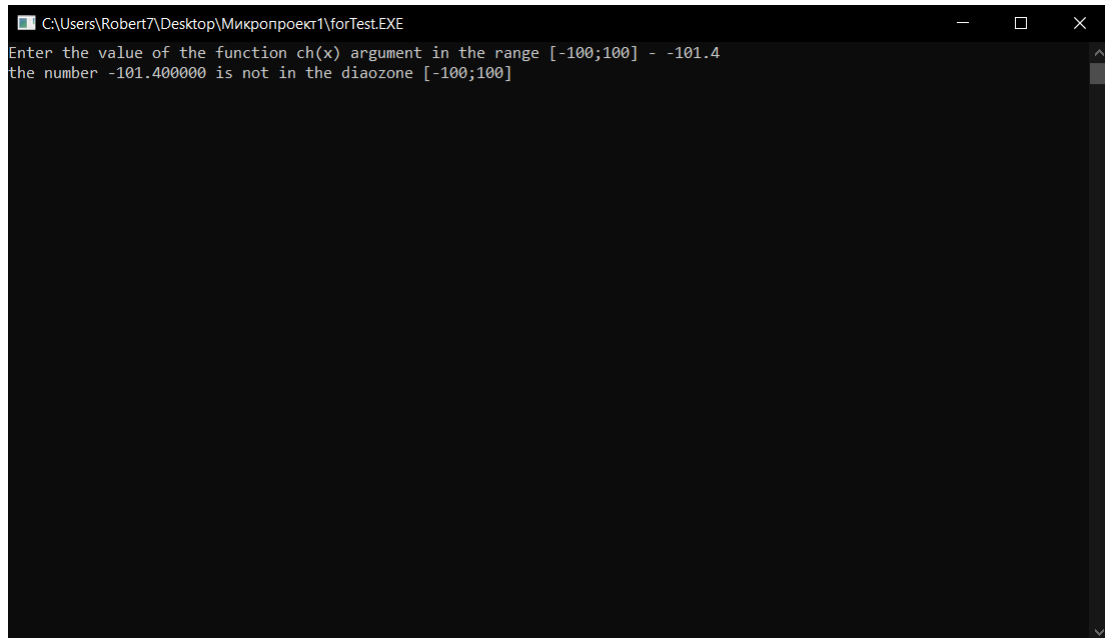


Рисунок 2

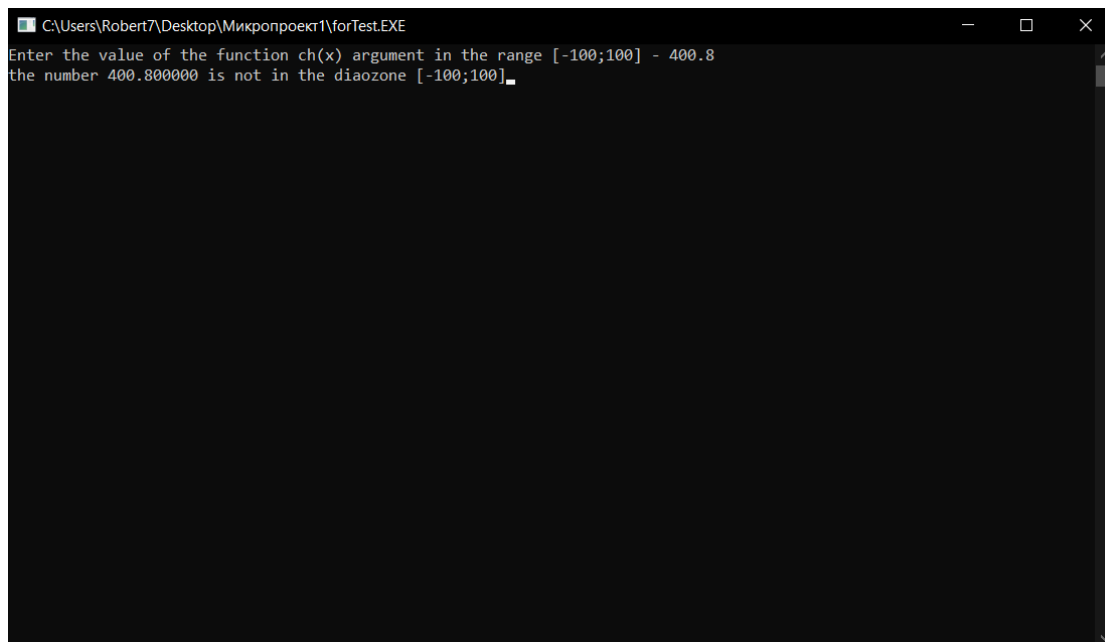
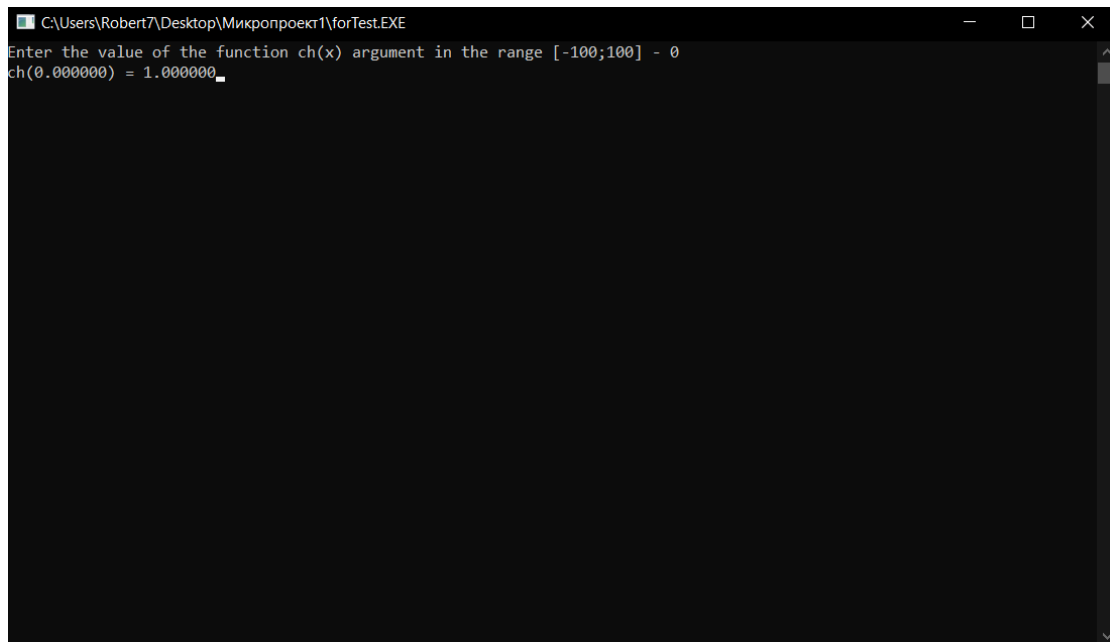


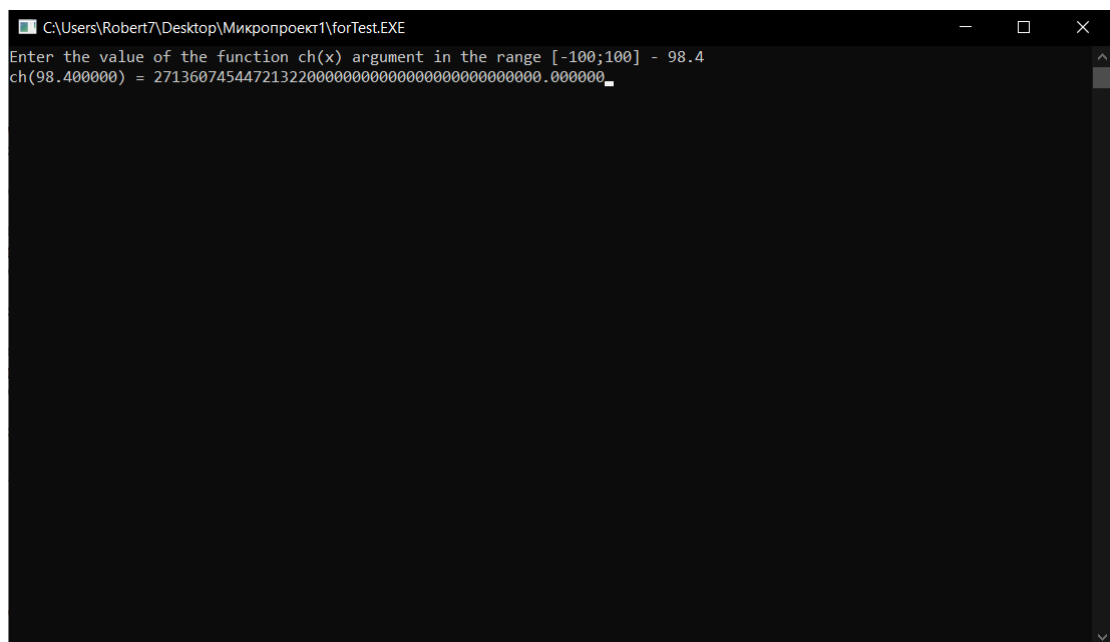
Рисунок 3

Корректная работа программы(рисунки 4-6)



```
C:\Users\Robert7\Desktop\Микропроект1\forTest.EXE
Enter the value of the function ch(x) argument in the range [-100;100] - 0
ch(0.000000) = 1.000000_
```

Рисунок 4



```
C:\Users\Robert7\Desktop\Микропроект1\forTest.EXE
Enter the value of the function ch(x) argument in the range [-100;100] - 98.4
ch(98.400000) = 2713607454472132200000000000000000000000000000000000000.000000_
```

Рисунок 5

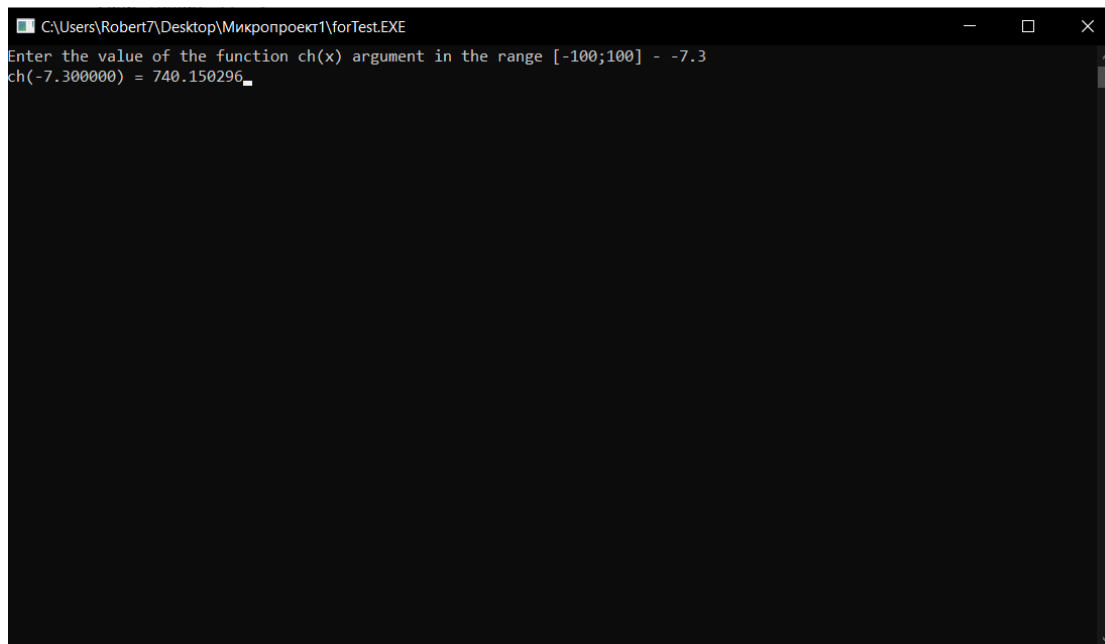


Рисунок 6

На проделанных тестах(от -100 до 100) результат вычислений отличается от действительного не более 0,01%(требовалось не более 0,05%)

Список используемых источников

1. Википедия (2019) “Гиперболические функции”
(https://ru.wikipedia.org/wiki/Гиперболические_функции#Разложение_в_степенные_ряды)
2. А.И. Легалов “Архитектура ВС. Примеры архитектур уровня набора команд (x86)”
(<http://www.softcraft.ru/edu/comparch/lect/03-archexamples/>)
3. А.И. Легалов “Разработка программ на ассемблере. Использование сопроцессора с плавающей точкой.”
(<http://www.softcraft.ru/edu/comparch/>)
4. Справочник по командам процессоров X86 (32-х разрядная архитектура) (<http://ccfit.nsu.ru/~kireev/lab2/lab2com.htm>)
5. Лекция 9. “Команды безусловного и условного переходов в языке Ассемблер” (<https://sites.google.com/site/sistprogr/lekci1/lek9>)
6. “Гиперболические функции” (<https://planetcalc.ru/1116/>)

7. “Расчет процентной погрешности”
(<https://ciox.ru/calculate-percentage-error>)