Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет

“Высшая школа экономики”»

###### **Факультет компьютерных наук**

Программная инженерия

Исполнитель: Мурзабеков Султан

**МИКРОПРОЕКТ**

Работа студента 2 курса бакалавариата группы БПИ-195

по предмету «Архитектура вычислительных систем»

Преподаватель:

Доктор технических наук,

Профессор

Легалов А. И

**Москва 2020**

**Задание:**

Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,05% значение функции ln(1-x) для заданного параметра x (использовать FPU).

**Решение.**

Если функция f(x) имеет на некотором интервале, содержащем точку а, производные всех порядков, то к ней может быть применена формула Тейлора[1]:

Разложить в степенной ряд функциюf(x) = ln(1-x). Найдем значения функции и ее производных при х=0, f(x)=ln(1-x), f(0)=0

Таким образом, последовательность

Подставляя полученные значения производных в формулу ряда Тейлора, получим:

или:

Функция ln(1-x) может быть разложена в степенной ряд Тейлора:

Для получения общего члена ряда a удобнее будет в отдельной переменной накапливать степени х, умножая эту переменную на х при каждой итерации, а для знаменателя завести отдельную переменную, увеличивающуюся на единицу при каждой итерации. А очередной член ряда получать деля первую переменную на вторую при каждой итерации.

Для проверки полученной суммы степенного ряда следует вычислить точное значение натурального логарифма. Но поскольку в наборе команд FPU отсутствует такая команда, но присутствует команда FYL2X для вычисления логарифма по основанию 2, для вычисления ln(х) можно воспользоваться формулой замены основания логарифма, приведенной в [2]:

log2e – константа, загружается в сопроцессор командой FLDL2E.

Напишем сначала программу, вычисляющую с помощью степенного ряда значение функции ln(1-x) на языке С++, чтобы «обработать» алгоритм.

Текст программы приведен ниже.

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

double x, s=0,eps= 0.0005;

double p = 1,a;

int c = 0;

cout << "x = ";

cin >> x;

do

{

p \*= x;

c++;

a = p / c;

s -= a;

} while (fabs(a) >= eps);

cout << s << endl;

cout << log(1-x) << endl;

system("pause");

return 0;

}

Итерационный цикл завершается когда очередной член суммы по модулю будет меньше заданной точности. Точность 0,05% соответствует абсолютной величине 0,0005.

Теперь, когда алгоритм отлажен, и результат совпадает с точным значением (в пределах заданной погрешности), перепишем данную программу на ассемблере для компилятора FASM.

Для ввода исходных данных воспользуемся функциями стандартной библиотеки си:

char \*gets(char \*str)

Функция gets() считывает символы из stdin и помещает их в массив символов, на который ука­зывает str. Символы считываются до тех, пока не встретится новая строка или EOF. Символ «но­вая строка» не делается частью строки, а транслируется в нулевой символ, завершающий строку.

Чтобы преобразовать строку в действительное значение, применим функцию sscanf, которая распознает и считывает​​ данные​​ по​​ заданному​​ шаблону из строки.​​

int sscanf(char \*buf, const char \*format, arg-list);

Она​​ имеет​​ следующие​​ аргументы:

● buf ​​ —​​ указатель​​ на​​ символьный​​ буфер​​, подлежащий считыванию;

● format — указатель на C-строку, содержащую формат результата;

● остальные​​ аргументы​​ —​​ данные,​​ подлежащие​​ форматированию.

Данная функция возвращает целое значение, которое представляет собой количество корректно распознанных значений из format, что позволяет реализовать проверку корректности ввода.

Так как у данной функции количество аргументов не постоянно, то она использует соглашение вызова cdecl, которое отличается от stdcall тем, что очистку стека от аргументов выполняет вызывающая функция, а не  
вызываемая.

Для вывода результата воспользуемся функцией printf, которая форматирует​​ данные​​ по​​ заданному​​ шаблону и выводит их на консоль.

int printf (const char \* format, ... );

​​ Она​​ имеет​​ следующие​​ аргументы:

● format — указатель на C-строку, содержащую формат результата;  
● остальные​​ аргументы​​ —​​ данные,​​ подлежащие​​ форматированию.

Так как у данной функции количество аргументов не постоянно, то она использует соглашение вызова cdecl, которое отличается от stdcall тем, что очистку стека от аргументов выполняет вызывающая функция, а не  
вызываемая.

Для завершения работы программы выполним вызов функции ExitProcess,  
которая​​ имеет​​ один​​ аргумент​​ —​​ код​​ завершения​​ работы​​ программы.

VOID ExitProcess(

UINT uExitCode // код выхода для всех потоков

);

Входной параметр x считывается из консоли (клавиатуры). Результат работы выводится на консоль (экран). Программу можно разбить на следующие функции:

Главная функция программы. В ней выполняет ввод исходных данных с проверкой их корректности, вызываются функции myln и log и выводятся на консоль результаты выполнения данных функций.

**double myln(double x,double eps);**

myln – функция вычисления ln(1-x) с помощью степенного ряда с заданной точностью eps. Вызывается по соглашению cdecl. Имеет локальные переменные:

c – константа, на которую делим

p – степень х

a – очередное слагаемое ряда

Результат возвращается в ST(0).

**double log(double x);**

log – функция точного вычисления ln(1-x). Вызывается по соглашению cdecl. Результат возвращается в ST(0).

Текст программы приведен ниже:

format PE Console ; 32-разрядная консольная программа WINDOWS EXE

entry start ; точка входа

include 'include\win32a.inc'

section '.idata' import data readable ; секция импортируемых функций

library kernel,'kernel32.dll',\

user,'user32.dll',\

msvcrt,'msvcrt.dll'

import kernel,\

ExitProcess,'ExitProcess'

import msvcrt,\

sscanf,'sscanf',\

gets,'gets',\

\_getch,'\_getch',\

printf,'printf'

section '.data' data readable writeable ; секция данных

x1 dq ? ;введенное значение

eps1 dd 0.0005 ;точность 0.05%

;константы

msg1 db 'Enter x (-1<=x<1): ',0

msg2 db 'Wrong number.',13,10,0

fmt1 db '%lf',0

msg3 db 'Teylor row = %lg',13,10,0

msg4 db 'Calculated ln(1-x) = %lg',13,10,0

buf db 256 dup(0)

section '.code' code readable executable ; секция кода

start: ; точка входа в программу

ccall [printf],msg1 ;вывести сообщение

ccall [gets],buf ;ввод строки с консоли

ccall [sscanf],buf,fmt1,x1 ;преобразовываем введенную строку в число

cmp eax,1 ;если преобразование удалось,

jz m1 ;продолжить

ccall [printf],msg2 ;вывести сообщение об ошибке

jmp start ;начать заново

m1: fld [x1] ;x

fld1 ;1

fcompp ;сравнить 1 с введенным числом

fstsw ax ;записать флаги сопроцессора в ах

sahf ;перенести их в флаги процессора

jbe start ;если 1<=x, начать заново

fld [x1] ;x

fld1 ;1

fchs ;-1

fcompp ;сравнить -1 с введенным числом

fstsw ax ;записать флаги сопроцессора в ах

sahf ;перенести их в флаги процессора

ja start ;если -1>x, начать заново

fld [eps1] ;точность вычисления

sub esp,8 ;выделить в стеке место под double

fstp qword [esp] ;записать в стек double число

fld qword [x1] ;введенное значение

sub esp,8 ;выделить в стеке место под double

fstp qword [esp] ;записать в стек double число

call myln ;Вычислить myln(x,eps)

add esp,16 ;удалить переданные параметры

sub esp,8 ;передать сумму ряда

fstp qword [esp] ;функции через стек

push msg3 ;формат сообщения

call [printf] ;сформировать результат

add esp,12 ;коррекция стека

fld qword [x1] ;введенное значение

sub esp,8 ;выделить в стеке место под double

fstp qword [esp] ;записать в стек double число

call log ;Вычислить log(x)

add esp,8 ;удалить переданные параметры

sub esp,8 ;передать точное значение логарифма

fstp qword [esp] ;функции через стек

push msg4 ;формат сообщения

call [printf] ;сформировать результат

add esp,12 ;коррекция стека

ccall [\_getch] ;ожидание нажатия любой клавиши

ex: stdcall [ExitProcess], 0;выход

;double myln(double x,double eps)

;вычисление ln(1-x) (x) с точностью eps

;соглашение вызова cdecl

myln:

push ebp ;создать кадр стека

mov ebp,esp

sub esp,14h ;создание локальных переменных

;локальные переменные

c equ ebp-14h ;конствнта, на которую делим

p equ ebp-10h ;степень х

a equ ebp-8h ;значение очередного слагаемого суммы

;переданные функции параметры

x equ ebp+8h

eps equ ebp+10h

fld1

fstp qword [p] ;p = 1

mov dword [c],0 ;c = 0

fldz ;s=0

lp1:

;p \*= x;

fld qword [p] ;накапливаем степень слагаемого

fmul qword [x]

fst qword [p]

;c++

inc dword [c] ;увеличить консанту, на которую делим

;a = p / c

fidiv dword [c] ;получили очередное слагаемое

fst qword [a] ;сохранить его

;s -= a;

fsubp st1,st

fld qword [a] ;a

fabs ;|a|

fcomp qword [eps] ;сравнить |a| c eps

fstsw ax ;перенести флаги сравнения в ах

sahf ;занести ah в флаги процессора

jnb lp1 ;если |a|>=eps, продолжить цикл

leave ;эпилог функции

ret

;double log(double x)

;точное вычисление ln(x)

;соглашение вызова cdecl

log:

push ebp ;создать кадр стека

mov ebp,esp

fld1 ;1

fsub qword [ebp+8] ;1-x

FLD1 ;1

fxch st1 ;поменять местами 1 и х

FYL2X ;вычислить 1\*log2(x)

FLDL2e ;Загрузить константу log2(e)

fdivp st1,st ;log2(x)/log2(e)=ln(x)

pop ebp ;эпилог функции

ret

Результат выполнения программы приведен на рисунке 1.

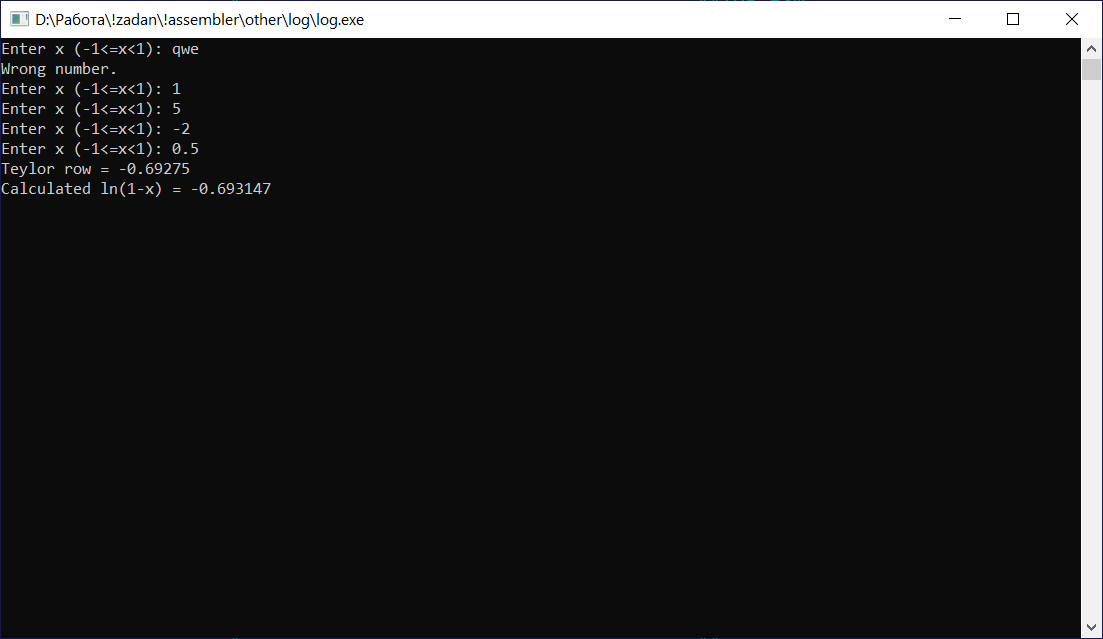


Рисунок 1 ­– Результат выполнения программы

Как видно из рисунка 1, программа обрабатывает некорректный ввод, недопустимые значения аргумента. Погрешность вычисления соответствует заданной.

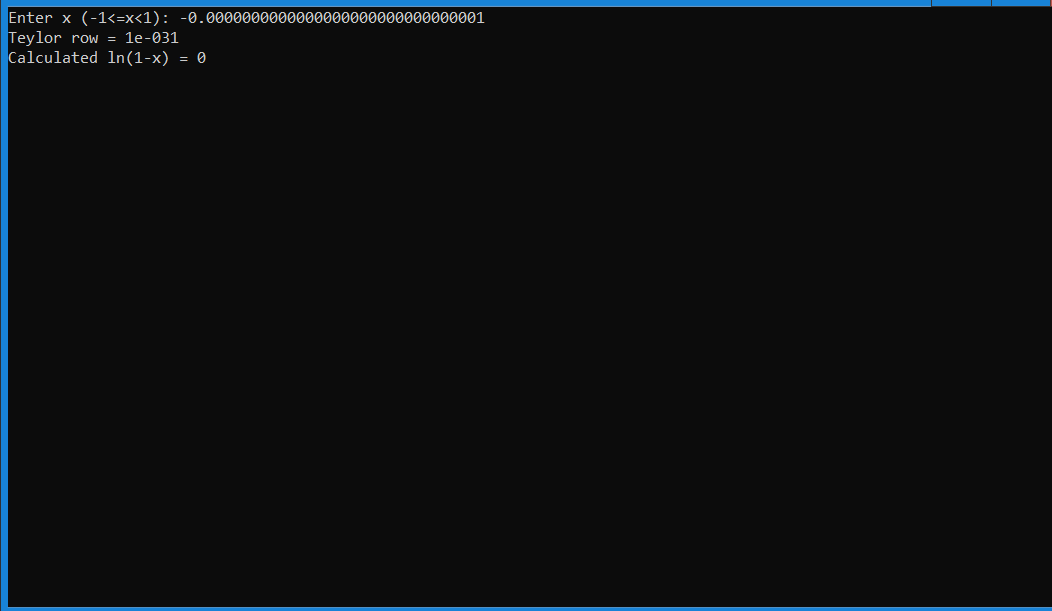


Рисунок 2 ­– Тест №2

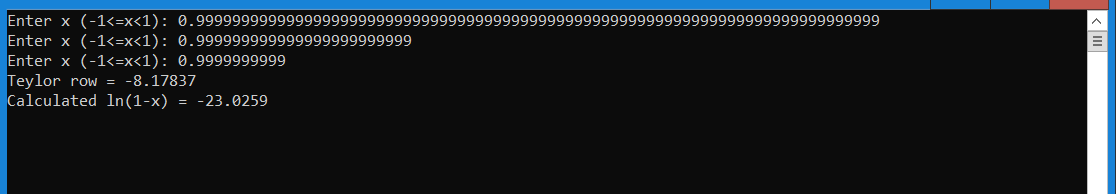


Рисунок 3 ­– Тест №3

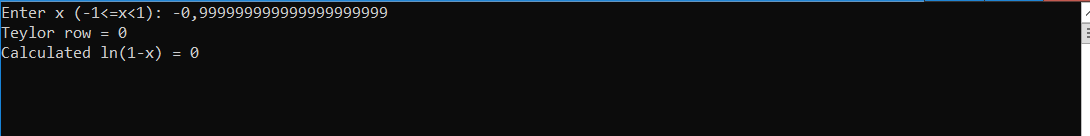


Рисунок 4 ­– Тест №4



Рисунок 5 ­– Тест №5

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 ­– Тест №6

**Список использованных источников**

1 <https://math.semestr.ru/math/taylor.php>

2 <https://ru.wikipedia.org/wiki/Логарифм>

3 Юров В., Хорошенко С. Assembler: учебный курс – СПб: Питер Ком, 1999. – 672с.: ил.

4 Кип Р.И. Язык ассемблера для процессоров Intel. М.: Издательский дом “Вильямс”, 2005. – 912с.