

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

ВЫЧИСЛЕНИЕ ТАНГЕНСА МЕТОДОМ РЯДА ТЕЙЛОРА

Пояснительная записка

Выполнил:
Гусейнов Ульви,
Студент группы БПИ198

Москва
2020

Содержание

1. Текст задания	2
2. Применяемые расчетные методы.....	2
2.1. Теория решения задания	2
3. Тестирование программы	3
ПРИЛОЖЕНИЕ 1	5
Список литературы.....	5
ПРИЛОЖЕНИЕ 2	6
Код программы	6-9

1. Текст задания

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

2. Применяемые расчетные методы

2.1. Теория решения задания

Для нахождения и вычисления степенного ряда $\tan(x)$ необходимо использовать формулу ряда Тейлора(ряд Маклорена). Чтобы вычислить $\tan(x)$ необходимо найденный синус разделить на найденный косинус с помощью ряда(рис 1).

$$\begin{aligned}
 e^x &= 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots, |x| < \infty, \\
 \sin x &= \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + \frac{(-1)^{n+1} x^{2n-1}}{(2n-1)!} - \dots, |x| < \infty, \\
 \cos x &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + \frac{(-1)^{n+1} x^{2n}}{(2n)!} - \dots, |x| < \infty, \\
 \ln(1+x) &= \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \dots + \frac{(-1)^{n+1} x^n}{n} - \dots, x \in (-1; 1], \\
 (1+x)^\alpha &= 1 + \frac{\alpha}{1!} x + \frac{\alpha(\alpha-1)}{2!} x^2 + \frac{\alpha(\alpha-1)(\alpha-2)}{3!} x^3 + \dots + \frac{\alpha(\alpha-1)\dots(\alpha-n+1)x^n}{n!} + \dots, |x| < 1, \\
 \operatorname{arctg} x &= x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots + \frac{(-1)^{n-1} x^{2n-1}}{2n-1} - \dots, |x| \leq 1, \\
 \frac{1}{1-x} &= 1 + x + x^2 + \dots + x^n + \dots, |x| < 1, \\
 \frac{1}{1+x} &= 1 - x + x^2 - \dots + (-1)^n x^n + \dots, |x| < 1, \\
 \operatorname{sh} x &= x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2n-1}}{(2n-1)!} + \dots, |x| < \infty, \\
 \operatorname{ch} x &= 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^{2n}}{(2n)!} + \dots, |x| < \infty, \\
 \ln \frac{1+x}{1-x} &= 2 \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots + \frac{x^{2n-1}}{2n-1} + \dots \right), |x| < 1, \\
 \frac{1}{(1-x)^2} &= 1 + 2x + 3x^2 + \dots + (n+1)x^n + \dots, |x| < 1.
 \end{aligned}$$

Рисунок 1. формула ряда Тейлора(ряд Маклорена)

3. Тестирование программы

При запуске программы открывается консоль, выводится информация на текстовый запрос на ввод «X» для вычисления тангенса(рис. 2).

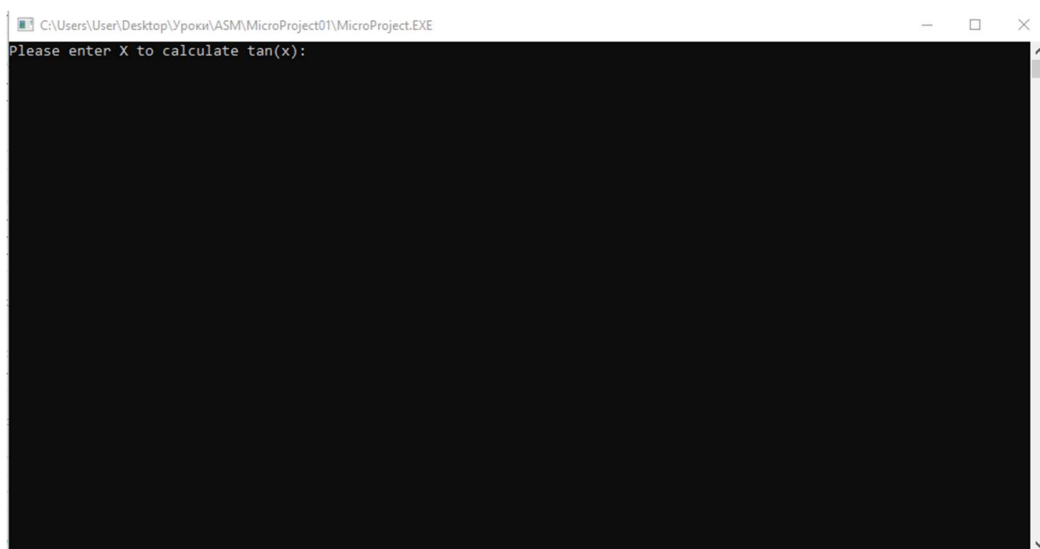


Рисунок 2. Запуск программы

Далее необходимо вписать значение «X» и нажать Enter(рис. 3).

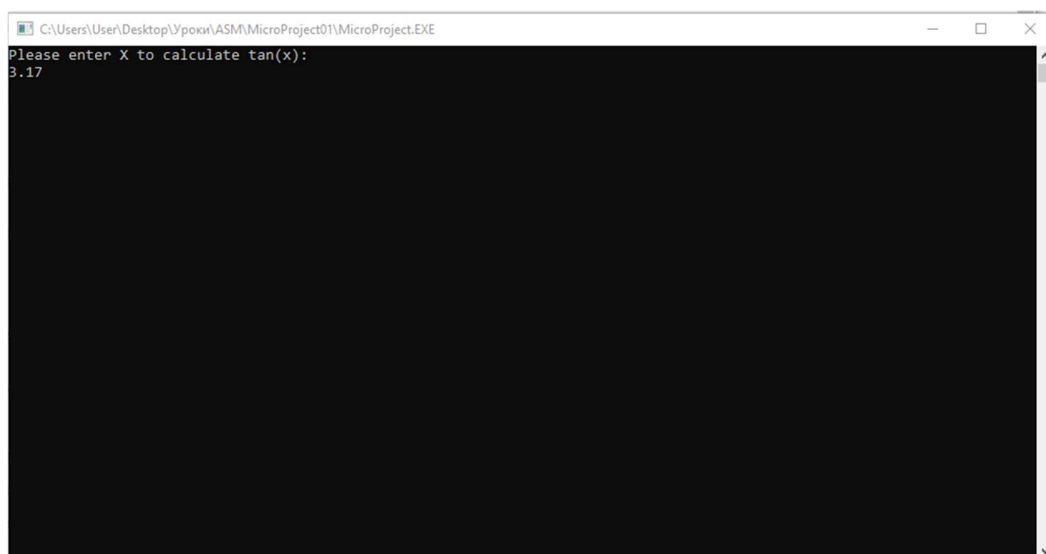
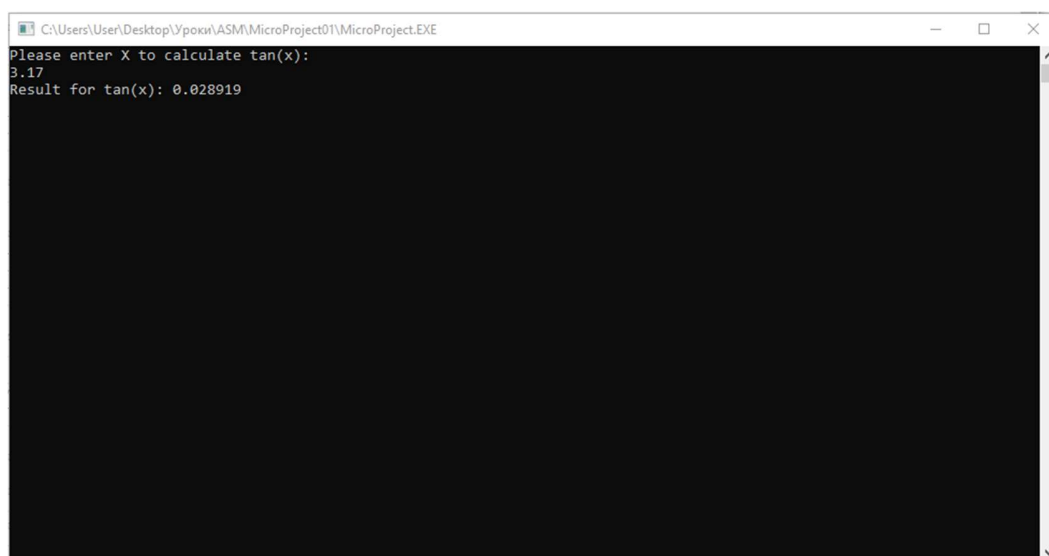


Рисунок 3. Пример работы при корректных входных данных

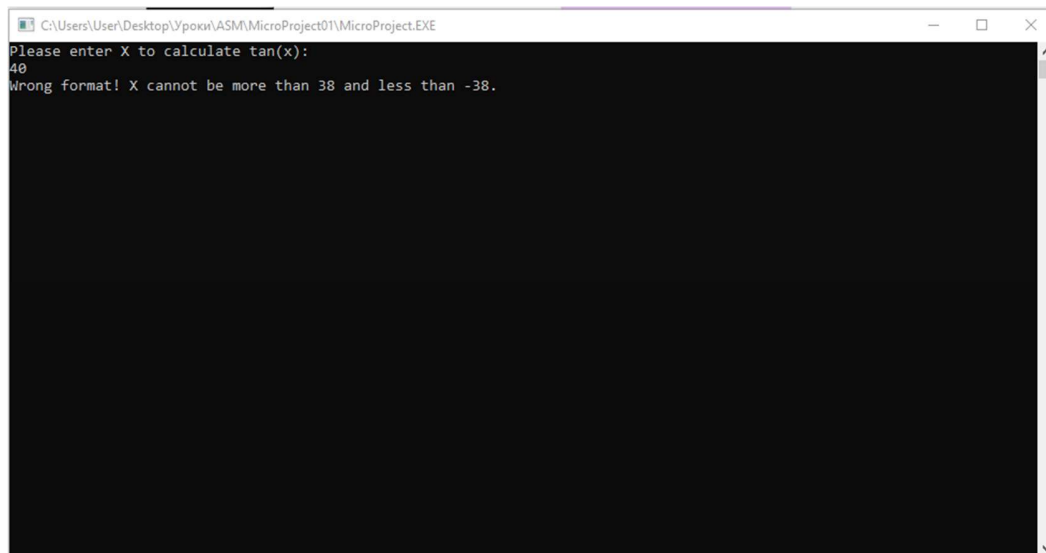
Далее будет выведен результат программы(рис4).



```
C:\Users\User\Desktop\Уроки\ASM\MicroProject01\MicroProject.EXE
Please enter X to calculate tan(x):
3.17
Result for tan(x): 0.028919
```

Рисунок 4. Результат программы

Если x больше 38 или меньше -38, будет выведено информация о некорректном вводе(рис. 4).



```
C:\Users\User\Desktop\Уроки\ASM\MicroProject01\MicroProject.EXE
Please enter X to calculate tan(x):
40
Wrong format! X cannot be more than 38 and less than -38.
```

Рисунок 4. Пример работы при некорректных входных данных

ПРИЛОЖЕНИЕ 1

Список литературы

1. Программирование на ассемблере на платформе x86-64 // Аблязов Р., 2011 г.
2. Руководство по препроцессору FASM. [Электронный ресурс] // URL: <https://wasm.in/threads/rukovodstvo-po-preprocessoru-fasm.31748/>, 2016 г.

Код программы

```

format PE console
include 'win32a.inc'
entry start

.data:
    strNotify db "Please enter X: ",0;строка для просьбы X
    strFloat db "%f, 0 ;строка для ввода-вывода в printf и scanf
    strPrintResut db 'Result of tan: %f,0;строка вывода результата
    x dd ?          ;переменная для X
    sin dd ?        ;переменная для вычисленного синуса
    cos dd ?        ;переменная для вычисленного косинуса
    tan dq ?        ;тангенс
    ACC dd 0.001     ;константа для точности
    Cn1 dd -1.0      ;константа для -1f
    C2 dd 2.0        ;константа 2f
    i dd ?          ;переменная для цикла
    NULL = 0         ;константа для ExitProcess

.code:
start:
    ;вызов вывода просьбы ввести X
    push strNotify
    call [printf]

    ;ввод числа, число сохраняем в x
    push x
    push strFloat
    call [scanf]

    ;вызов функции вычисления тангенса
    call TanFunc

    ;вывод тангента
    push dword[tan+4] ;вывод через double
    push dword[tan]   ;заполняем ячейку
    push strPrintResut
    call [printf]

    ;ожидание нажатия любой клавиши чтобы приложение не закрылось сразу
    call [getch]

    ;Для закрытия программы без ошибок
    push NULL
    call [ExitProcess]

;функция вычисления тангенса
TanFunc:
    ;вызов функции вычисления синуса
    call SinFunc
    ;вызов функции вычисления косинуса
    call CosFunc

    ;берем синус
    fld dword[sin]

```

```

;делим на косинус
fdiv dword[cos]
;выделяем результат в переменную tan
fstp qword[tan]
ret

```

;функция вычисления синуса
SinFunc:

```

;загружаем точность
fld dword[ACC]
;n=x
fld dword[x]
;ячейка для суммы
fldz
;i=1
mov [i],1
;начало цикла
.loop:
;сумма плюс n
fadd st0,st1
;заполняем X
fld dword[x]
;умножаем X на себя
fmul st0,st0
;умножаем на константу -1
fmul dword[Cn1]
;заполняем i
fild [i]
;умножаем на 2
fmul dword[C2]
;дублируем 2*i
fld st0
;заполняем 1
fld1
;добавляем 1 к 2*i, после очищаем 1
faddp st1,st0
;умножаем 2*i на 2*i+1
fmulp st1,st0
;делим -1*x*x на (2*i)*(2*i+1)
fdivp st1,st0
;умножаем результат на n
fmul st0,st2
;записываем результат в n, очищаем ячейку
fstp st2
;увеличиваем i на 1
inc [i]
;загружаем копию n
fld st1
;берем модуль
fabs
;сравниваем последнее число
fcomip st3
;если |n|>ACC то начинаем цикл заново
jae .loop
;заполняем ответ в sin
fstp dword[sin]
;очищаем занятые ячейки
fstp st0
fstp st0

```



```
ret
```

```
;функция вычисления косинуса
```

```
CosFunc:
```

```
;загружаем точность
```

```
fld dword[ACC]
```

```
;n=1
```

```
fld1
```

```
;ячейка для суммы
```

```
fldz
```

```
;i=1
```

```
mov [i],1
```

```
;начало цикла
```

```
.loop:
```

```
;сумма плюс n
```

```
fadd st0,st1
```

```
;заполняем X
```

```
fld dword[x]
```

```
fmul st0,st0
```

```
fmul dword[Cn1]
```

```
;заполняем I
```

```
fild [i]
```

```
;умножаем на 2
```

```
fmul dword[C2]
```

```
;дублируем 2*1
```

```
fld st0
```

```
;заполняем 1
```

```
fld1
```

```
;меняем знак единицы
```

```
fchs
```

```
faddp st1,st0
```

```
fmulp st1,st0
```

```
fdivp st1,st0
```

```
fmul st0,st2
```

```
;записываем результат в n
```

```
fstp st2
```

```
;увеличиваем i
```

```
inc [i]
```

```
fld st1
```

```
;берем модуль
```

```
fabs
```

```
;сравниваем число
```

```
fcomip st3
```

```
jae .loop
```

```
;выгружаем ответ в переменную cos
```

```
fstp dword[cos]
```

```
;очищаем занятые ячейки
```

```
fstp st0
```

```
fstp st0
```

```
ret
```

```
section '.idata' import data readable
```

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

```
msvcrt, 'msvcrt.dll'
```

```
include 'api\kernel32.inc'
```

```
import kernel,\
```

ExitProcess, 'ExitProcess'

```
import msvcrt,\nprintf, 'printf', \ngetch, '_getch', \nscanf, 'scanf'
```