

# Микропроект 1

## Пояснительная записка

### 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)

### Описание расчётного алгоритма программы:

$Rezult = \tan(n)$  if  $\{[abs(\tan(n)) - abs(\tan(n - 1))] < (abs(\tan(n)) * 0.001)\}$ ;

Где  $\tan(n) = \sin(n) / \cos(n)$ ;

$$\sin(n) = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \dots = \sum_{n=0}^{+\infty} \frac{(-1)^n * X^{2n+1}}{(2n+1)!};$$

$$\cos(n) = 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} + \dots = \sum_{n=0}^{+\infty} \frac{(-1)^n * X^{2n}}{(2n)!};$$

$$X = \frac{x}{abs(x)} * (abs(x) \% (2 * \pi)) = sign(x) * (|x| \bmod (2 * \pi)) ;$$

Изначально вычисляется  $X = (x \pm 2 * \pi k)$  для экономии памяти во время вычисления рядов.

После этого вычисляются приблизительные значения  $\sin(x)$  и  $\cos(x)$  относительно текущего  $n$  (номера текущего цикла (см. counter)) с помощью рядов Маклорена для  $\sin$  и  $\cos$ , затем вычисляется  $\tan(x) = \sin(x) / \cos(x)$  и сравнивается с предыдущим значением при  $n-1$ . В случае если разница между предыдущим значением и текущим  $\tan(x)$  не превышает 0.1% текущий тангенс является ответом, иначе повтор цикла при  $n+1$ . Роль  $n$  в программе выполняет переменная counter (см. Описание элементов программы).

### Список используемых источников для решения задачи:

Учебник fasm: <http://flatassembler.narod.ru/fasm.htm#2-1-13>

Система команд сопроцессора: <https://prog-cpp.ru/asm-coprocessor-command/>

Более подробное описание команд: <https://www.club155.ru/x86cmdfpu/>

Материалы с SoftCraft: <http://www.softcraft.ru/edu/comparch/practice/asm86/05-fpu/>

Учебные материалы с семинаров/лекций:

[https://drive.google.com/drive/folders/1tmsLo00jlsL2dSmxBuuFiWTv\\_GWTdPu](https://drive.google.com/drive/folders/1tmsLo00jlsL2dSmxBuuFiWTv_GWTdPu)

Собственные проекты в качестве шаблонов для некоторых фрагментов кода:

<https://github.com/MAGGen-hub/FASM-Projects-and-Tests>

## Описание файлов в папке проекта:

- 1) microproject1.docx – исходник для пояснительной записки.
- 2) microproject1.pdf – пояснительная записка.
- 3) microproject1.ASM – файл с исходным кодом программы, пригодным для компиляции и запуска.
- 4) microproject1.EXE – готовая скомпилированная программа.
- 5) calculate.inc – файл с макросом “calculate”.
- 6) screen.png – скриншот работы программы с проверкой значений

## Описание элементов программы:

- 1) Переменные из сектора Data:

Source:

```
StrX db 'Enter x in rad:', 0
StrCor db 'Your value: %lf is it correct? [y/n]:', 0
StrRez db 'Answer: %lf', 0
strRepeat db 10,13, 'Repeat? [y/n]:', 0
```

Description:

Строковые переменные размером в 1 байт каждая для вывода соответствующих строк в консоль по мере надобности.

Source:

```
enterStr db 10,13,0
scfprmlf db '%lf',0
```

Description:

Переменная формата для сканирования входного значения, и строка, имитирующая нажатие клавиши Enter.

x dq 0 ; - переменная размером 8 байт для считывания входного значения  
counter dd 2 ; - переменная счётчик (4 байта) для вычисления факториала и степени x.

fact dq ? ; - переменная (8 байт) для хранения факториала от counter  
powx dq ? ; - переменная (8 байт) для хранения x в степени counter  
sin dq ? ; - переменная (8 байт) для хранения sin(x) для текущего значения counter {n}

cos dq 1.0 ; - аналогична sin, только для cos  
tan dq ? ; - (8 байт) для вычисления tan(x) для текущего значения counter  
persent dq 0.001 ; - константа (8 байт) для хранения значения точности.

- 2) Методы (секции кода) из сектора Code:

Start: - точка входа в программу. Инициализирует FPU и задаёт значения некоторых переменных на случай повтора программы.

GetValue: - запрашивает у пользователя значение переменной  $x$  и считывает его.

CheckValue: - запрашивает у пользователя подтверждение правильности введённого им значения и даёт возможность вернуться к GetValue:.

Program: - “основная часть программы”, место с которого начинается обработка переменной  $x$ .

DivToPi: - “вызывается” из Program: если модуль  $x$  превышает  $2\pi$ .  
Используется для получения остатка от  $x/(2\pi)$ . (Находит  $X$  см. описание алгоритма программы)

RmvAbs: - “вызывается” из Program: если DivToPi: не требуется ( $x = X$  в этом случае)

LastPrepare: - идёт после DivToPi: и RmvAbs: выполняет подготовку оставшихся переменных для вычисления рядов:  $x = X$ ,  $\text{pow}x = X$ ,  $\sin = X$ ,  $\text{fact} = 1$ , в верхушку стека FPU записывается -1 для контроля знака в рядах.

GetCos: - начало цикла для вычисления рядов, вычисляет ряд  $\cos$  от текущего значения counter (прибавляет эл-т к уже существующему ряду в  $\cos$ ) (counter ++).

GetSin: - вычисляет ряд  $\sin$  от текущего значения counter (прибавляет эл-т к уже существующему ряду в  $\sin$ ) (counter ++).

GetTan: - получает  $\tan$  от текущего counter и определяет, стоит ли продолжать вычисления или желаемая точность уже достигнута. В случае её достижения переходит к Finalize: иначе запускает GetCos: снова. В этой секции происходит смена знака в верхушке стека FPU поставленного ранее в секции LastPrepare:.

Finalize: - завершает вычисление  $\tan$  и выводит результат на экран консоли. Чистит стек FPU от оставшихся в нём значений, а также предлагает пользователю перезапустить программу для вычисления нового значения, в случае положительного ответа переходит к Start:

Exit: - выход из программы. Вызывает [ExitProcess].

### 3) Описание макроса calculate

Данный макрос был создан только для сокращения кода в GetCos: и GetSin: так как большая часть вычислений в них одинаковая.

## Приложение

### Код программы

format PE console 4.0

entry Start

include 'win32a.inc'

include 'calculate.inc'

```
;-----  
;  
; Data:  
;-----
```

section '.data' data readable writable

;programm strings

StrX db 'Enter x in rad:', 0

StrCor db 'Your value: %lf is it correct? [y/n]:', 0

StrRez db 'Answer: %lf', 0

strRepeat db 10,13, 'Repeat? [y/n]:', 0

enterStr db 10,13,0

scfprmlf db '%lf',0

;usable variables:

x dq 0 ; starter value

counter dd 2 ; for pow and fact string

fact dq ?; for factorial

powx dq ?; x in conter pow

sin dq ?; sinus of counter

cos dq 1.0; cosinus of counter

tan dq ?; tangens of counter -1

percent dq 0.001 ; only for 0.1%

```
;-----  
;  
; Code:  
;-----
```

section '.code' code readable executable

Start: ; program start

fini ;init FPU

fld1 ; init variabler (for repeat function)

fstp [cos]

mov eax,2

mov [counter],eax

fildz ; add "zero"

GetValue:

fst [x] ;x to zero

push StrX

call [printf]

push x

push scfprmlf

call [scanf]

CheckValue: ;user check value

invoke printf, StrCor,dword[x],dword[x+4]

invoke printf, enterStr

call [getch] ; user input

cmp eax, 121 ;121 <=> y

```

je Program
cmp eax, 173 ;173 <=> í
je Program
jmp GetValue ; not y or í key

```

;------

; region start: main program

Program:

```

fstp [tan] ; (delete st(0) throw useles trash into tan till it unused)
fld [x] ; x to st(0) (zero deleted)
fabs ; st(0) = abs(st0)
fldpi ; pi to st(0) (x to st(1))
fild [counter] ; 2 ro st(0)
fmulp ; st(0) = 2*pi st(1) = x
fcomi st1 ; cmp 2*pi,x
ja RmvAbs ; if (x < 2*pi) jump to normal prog

```

DivToPi: ; if x >= 2\*pi div to 2\*pi

```

fxch ; x <-> 2*pi (st(0)=x,st(1)=2*pi)
fprem ; get st(0) = st(0) - Q*st(1) (get new x (x%(2*pi)))
fxch ; st(0) <-> st(1)
fstp [tan] ; throw away 2*pi from stack (st(0) = x%(2*pi)) (stack (without st(0))is empty) <tan still unused>
fld [x] ;fld real x st(0)=x st(1)= abs(x%2*pi)
fcomip st1; remove st(0)=x from stack (only st(0) = abs(x%(2*pi)))
jae LastPrepare ; if (st(0) >= st(1)) => (x >= abs(x%(2*pi))) => x > 0 => jmp to normal program without RmvAbs
; else (x < 0) change sign
fchs ; st(0)= -st(0) => abs(x%(2*pi)) = -abs(x%(2*pi)) (case of x sign)
jmp LastPrepare ; sign changed => go to normal prog

```

RmvAbs:

```

fstp [tan]; (i don't use fincstp case it ALWAYS broke my program)
fstp [tan];clear stack
fld[x];if this code is runing => abs(x) < 2*pi => x is normal for program (st(0)= right x)

```

LastPrepare: ;iterating all variables

```

fst [x] ; now x = x%(2*pi)
fst [powx] ; powx = x^1
fstp [sin] ; start sin string with x (stack empty)
fld1 ; to control the sign
fst [fact]
fchs ; st(0) = -1

```

```

fld[sin]
fld[cos]
fdivp
fstp [tan]; first "tangens"

```

;LOOP starts here

GetCos: ; getting cosinus value of curent counter value (input => x < 2\*pi for easy calculating)

calculate

```

fld [cos]
faddp ; st(1)= cutent sign; st(0) = cur_cos*cur_sign + last_cos_value
fstp [cos]

```

GetSin: ; getting sinus value of curent counter value

calculate

```

fld [sin]

```

```
faddp ; st(1)= cutent sign; st(0) = cur_sin*cur_sign + last_sin_value
fstp [sin]
```

GetTan: ; getting tangens value of curent counter value to compere with last value

```
fld [sin]
fld [cos]
fdivp ;get curent tangens st(0) = sin/cos st(1) = cur_sign (st(0) = new tan)

fld [tan] ;old tan
fsub st0, st1 ; difference between new tan and old tan st(0) = dif st(1) = new tan st(2) = cur_sign
fabs; abs (diff)
fld [persent]
fmul st0,st2 ;0.001*new_tan st(0) = needed diff st(1) = diff st(2) = new_tan st(3) = cur_sign
fabs; abs(needed diff)

fcomp st1 ;
jae Finalize; if (needed_diff >= diff) jump to finish (LOOP ends here)
;else
fstp [tan];trash (diff)
fstp [tan]; new tangens (st(0)= cur_sign)
```

```
fchs ; st(0) = cur_sign (change cur sign)
```

```
jmp GetCos
```

```
; region end: main program
```

```
;-----
```

Finalize: ; get tangens value and go to exit

```
fstp [tan]; trash (diff)
fstp [tan]; new tangens
fstp [counter]; clear stack from any values
```

```
invoke printf, StrRez,dword[tan],dword[tan+4] ;answer
```

```
invoke printf, strRepeat ; repeat function
```

```
invoke printf, enterStr
```

```
call [getch]
```

```
cmp eax, 121 ;121 <=> y
```

```
je Start
```

```
cmp eax, 173 ;173 <=> í
```

```
je Start
```

Exit: ; exit form program

```
push 0
```

```
call [ExitProcess]
```

```
;-----
```

```
; Import:
```

```
;-----
```

section '.idata' import data readable

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

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

```
import kernel,\
```

```
ExitProcess, 'ExitProcess'
```

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