НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

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

МИКРОПРОЕКТ

ВАРИАНТ 4

Пояснительная записка

Листов 16

Выполнил:
Асатрян Эмин,
студент гр. БПИ198

Москва
2020

СОДЕРЖАНИЕ

1. ТЕКСТ ЗАДАНИЯ 3
2. ОПИСАНИЕ ВХОДНЫХ ДАННЫХ 4
3. ОПИСАНИЕ ВЫХОДНЫХ ДАННЫХ 5
4. ОПИСАНИЕ ПРИМЕНЯЕМЫХ РАССЧЕТНЫХ МЕТОДОВ 6
4.1. Переменные 6
4.1.1. Исходные данные 6
4.1.2. Промежуточные данные 6
4.2. Теория решения основной задачи 7
4.3. Методы 9
5. ТЕСТИРОВАНИЕ ПРОГРАММЫ 10
5.1. Корректные входные данные 10
5.2. Некорректные входные данные 11
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12
ПРИЛОЖЕНИЕ 1. Код программы 13

1. ТЕКСТ ЗАДАНИЯ

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

2. ОПИСАНИЕ ВХОДНЫХ ДАННЫХ

На вход программе подается вещественное число x – угол в радианах, косинус которого нужно посчитать. Вещественное число x должно лежать в отрезке [-5π, 5π]. Такое ограничение наложено во избежание переполнения при работе с вещественными числами в программе.

3. ОПИСАНИЕ ВЫХОДНЫХ ДАННЫХ

В качестве результата программа выводит две строки:

1) В первой строке содержится результат подсчета с помощью степенного ряда значения косинуса введенного угла в виде следующий строки: “Computed with mathematical series cos(<угол в радианах>) = <значение косинуса>”.

2) Во второй строке содержится результат подсчета с помощью FCOS значения косинуса введенного угла в виде следующий строки: “Computed with FCOS cos(<угол в радианах>) = <значение косинуса>”.

4. ОПИСАНИЕ ПРИМЕНЯЕМЫХ РАССЧЕТНЫХ МЕТОДОВ

4.1. Переменные

4.1.1. Исходные данные

1) floatFormat = '%lf' – Тип db - Строка-формат вещественного числа

2) enterXText = 'Enter x in [%dpi, %dpi]: ' – Тип db - Текст ввода угла

3) cosXResultText = 'Computed with mathematical series cos(%lf) = %lf' - Тип db - Текст результата подсчета с помощью рядов значения косинуса

4) realCosXResultText = 'Computed with FCOS cos(%lf) = %lf' - Тип db - Текст результата подсчета с помощью FCOS значения косинуса

5) xOutOfBoundsText = 'Entered x is not in allowed bounds...' - Тип db - Текст сообщения пользователю выхода за допустимые границы введенного угла

6) percentMultiplyer = 0.001 – Тип dq - Процент точности вычисления, переведенный в коэффициент пропорциональности

7) higherBoundPI = 5 – Тип dd - Коэффициент, задающий верхнюю границу (верхняя граница = PI * higherBoundPI)

8) lowerBoundPI = -5 – Тип dd - Коэффициент, задающий нижнюю границу (нижняя граница = PI * lowerBoundPI)

4.1.2. Промежуточные данные

1) sum = 0.0 – Тип dq - Текущая сумма ряда

2) prevTerm = 1.0 – Тип dq - Предыдущий член ряда

3) n = 0 Тип dq - Текущее значение итератора

4) x – Тип dq - Введенный угол в радианах

5) intTmp – Тип dd - Временная переменная для хранения целых чисел

6) floatTmp – Тип dq - Временная переменная для хранения вещественных чисел

4.2. Теория решения основной задачи

Основная задача заключается в применении следующей формулы разложения функции cos(x) в степенной ряд:

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

Для этого была реализована процедура, реализующая бесконечный цикл с проверкой на условие выхода из цикла в конце этого цикла.

Изначально sum = 0, n = 0, а prevTerm (если учесть, что n = 0) = $(-1)^0 \frac{x^0}{0!} = 1$.

Это значит, что уже подсчитан первый член ряда, поэтому цикл должен начинаться с увеличения n на единицу и обновления текущей суммы ряда. То есть мы инкрементируем n, и прибавляем к sum число prevTerm:

```
sumLoop:
inc [n]
FLD [sum]
FADD [prevTerm]
FSTP [sum]
```

Далее, в том же цикле нам нужно поместить в переменную prevTerm значение нового члена ряда, соответствующего обновленному текущему значению n. Наиболее эффективно это представляется сделать следующим образом – не высчитывать каждый раз член, а использовать уже имеющееся значение предыдущего члена prevTerm. Во первых, при увеличении n на единицу, из-за наличия множителя $(-1)^n$, нам нужно умножить prevTerm на -1 (поменять знак). Потом, поскольку степень x в формуле равна 2n, то при увеличении n на единицу нам придется умножить prevTerm на x^2. Ну и наконец, чтобы при текущем n из $\frac{1}{(2(n-1))!} = \frac{1}{(2n-2)!}$ получить $\frac{1}{(2n)!}$, нужно поделить prevTerm на (2n -1) и на 2n:

```
FLD [prevTerm]
FCHS
FMUL [x]
FMUL [x]
mov eax, [n]
imul eax, 2
mov [intTmp], eax
FIDIV [intTmp]
dec [intTmp]
FIDIV [intTmp]
FSTP [prevTerm]
```

Таким образом, в prevTerm будет лежать новый член ряда.

Теперь, нужно проверить, не выполнилось ли условие выхода из цикла, т.е. условие abs(prevTerm) <= percentMultiplyer * abs(sum). Если условие выполнилось, то выходим из цикла с помощью команды условного перехода (новый член не войдет в сумму, так как нужная точность уже достигнута):

```
FLD [percentMultiplyer]
```

```
FMUL [sum]
FABS
FLD [prevTerm]
FABS
FCOMPP
FSTSW ax
SAHF
jbe sumLoopEnd
```

Если условие выхода не выполнено, начинаем следующую итерацию цикла с помощью команды безусловного перехода (и в его начале к sum будет прибавлен только что подсчитанный prevTerm).

4.3. Методы

1) readX – процедура без входных параметров - вводит угол от пользователя

2) countCosX - процедура без входных параметров – рассчитывает косинус введенного угла с помощью степенного ряда

3) printCosX - процедура без входных параметров – выводит рассчитанный в методе countCosX косинус введенного угла

4) printRealCosX - процедура без входных параметров – рассчитывает косинус введенного угла с помощью FCOS и выводит его

5. ТЕСТИРОВАНИЕ ПРОГРАММЫ

5.1. Корректные входные данные

1) Ввод 0 (рис. 1)

```
Enter x in [-5pi, 5pi]: 0
Computed with mathematical series cos(0.000000) = 1.000000
Computed with FCOS cos(0.000000) = 1.000000
```

Рисунок 1. Результат ввода числа 0.

2) Ввод π (рис. 2)

```
Enter x in [-5pi, 5pi]: 3.141
Computed with mathematical series cos(3.141000) = -0.999900
Computed with FCOS cos(3.141000) = -1.000000
```

Рисунок 2. Результат ввода числа, приблизительно равному π.

3) Ввод -1 (рис. 3)

```
Enter x in [-5pi, 5pi]: -1
Computed with mathematical series cos(-1.000000) = 0.540278
Computed with FCOS cos(-1.000000) = 0.540302
```

Рисунок 3. Результат ввода числа -1.

4) Ввод верхней границы (рис. 4)

```
Enter x in [-5pi, 5pi]: 15.705
Computed with mathematical series cos(15.705000) = -1.000184
Computed with FCOS cos(15.705000) = -0.999996
```

Рисунок 4. Результат ввода числа, приблизительно равному 5π.

5) Ввод нижней границы (рис. 5)

```
Enter x in [-5pi, 5pi]: -15.705
Computed with mathematical series cos(-15.705000) = -1.000184
Computed with FCOS cos(-15.705000) = -0.999996
```

Рисунок 5. Результат ввода числа, приблизительно равному -5π.

5.2. Некорректные входные данные

1) Ввод числа, большего верхней границы (рис. 6)

```
Enter x in [-5pi, 5pi]: 16
Entered x is not in allowed bounds...
```

Рисунок 6. Результат ввода числа 16.

2) Ввод числа, меньшего нижней границы (рис. 7)

```
Enter x in [-5pi, 5pi]: -16
Entered x is not in allowed bounds...
```

Рисунок 7. Результат ввода числа -16.

3) Ввод строки (рис. 8)

```
Enter x in [-5pi, 5pi]: abcd
Computed with mathematical series cos(0.000000) = 1.000000
Computed with FCOS cos(0.000000) = 1.000000
```

Рисунок 8. Ввод строки “abcd”.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Разработка программ на ассемблере. Использование сопроцессора с плавающей точкой [Электронный ресурс] – URL: http://softcraft.ru/edu/comparch/practice/asm86/05-fpu (дата обращения: 31.10.20).
2. Условные и безусловные переходы [Электронный ресурс] – URL: http://osinavi.ru/asm/4.html (дата обращения: 31.10.20).
3. Команды сравнения FPU [Электронный ресурс] – URL: http://osinavi.ru/asm/FPUexpansion/5.html (дата обращения: 31.10.20).

ПРИЛОЖЕНИЕ 1. Код программы

```
format PE console

entry start

include 'win32a.inc'

; Секция данных
section '.data' data readable writable

    ; Строка-формат вещественного числа
    floatFormat db '%lf', 0

    ; Текст ввода угла
    enterXText db 'Enter x in [%dpi, %dpi]: ', 0

    ; Текст результата подсчета с помощью рядов значения косинуса
    cosXResultText db 'Computed with mathematical series cos(%lf) = %lf', 10, 0

    ; Текст результата подсчета с помощью FCOS значения косинуса
    realCosXResultText db 'Computed with FCOS cos(%lf) = %lf', 10, 0

    ; Текст сообщения пользователю выхода за допустимые границы введенного угла
    xOutOfBoundsText db 'Entered x is not in allowed bounds...', 10, 0


    ; Процент точности вычисления, переведенный в коэффициент пропорциональности
    percentMultiplyer dq 0.001

    ; Коэффициент, задающий верхнюю границу (верхняя граница = PI * higherBoundPI)
    higherBoundPI dd 5

    ; Коэффициент, задающий нижнюю границу (нижняя граница = PI* lowerBoundPI)
    lowerBoundPI dd -5


    ; Текущая сумма ряда
    sum dq 0.0

    ; Предыдущий член ряда
    prevTerm dq 1.0

    ; Текущее значение итератора
    n dd 0

    ; Введенный угол в радианах
    x dq ?

    ; Временная переменная для хранения целых чисел
    intTmp dd ?

    ; Временная переменная для хранения вещественных чисел
    floatTmp dq ?

    NULL = 0
```

```
; Секция кода
section '.code' code readable executable

; Точка входа программы
start:
        call readX
        call countCosX
        call printCosX
        call printRealCosX

; Ожидание нажатия клавиши для завершения работы программы
finish:
        invoke getch
        push NULL
        invoke ExitProcess

; Проедура ввода угла
proc readX

        ; Предлоение пользователю ввода и чтение
        invoke printf, enterXText, [lowerBoundPI], [higherBoundPI]
        add esp, 12
        invoke scanf, floatFormat, x
        add esp, 8

        ; Проверка на выход за верхнюю границу
        FLDPI
        FIMUL [higherBoundPI]
        FLD [x]
        FCOMPP
        FSTSW ax
        SAHF
        ja xOutOfBounds

        ; Проверка на выход за нижнюю границу
        FLDPI
        FIMUL [lowerBoundPI]
        FLD [x]
        FCOMPP
        FSTSW ax
        SAHF
        jb xOutOfBounds

        ret

; Обработка выхода за границы
xOutOfBounds:
        invoke printf, xOutOfBoundsText
        add esp, 4
        jmp finish

endp
```

```
; Процедура подсчета с помощью степенного ряда значения косинуса
proc countCosX

	; Инициализация сопроцессора
	FINIT

; Начало цикла последовательного суммирования членов ряда
sumLoop:
	; ++n, sum += prevTerm
	inc [n]
	FLD [sum]
	FADD [prevTerm]
	FSTP [sum]

	; prevTerm *= -1 * x * x / (2n) / (2n - 1)
	FLD [prevTerm]
	FCHS
	FMUL [x]
	FMUL [x]
	mov eax, [n]
	imul eax, 2
	mov [intTmp], eax
	FIDIV [intTmp]
	dec [intTmp]
	FIDIV [intTmp]
	FSTP [prevTerm]

	; Если выполнено abs(prevTerm) <= percentMultiplyer * sum, выходим из цикла
	FLD [percentMultiplyer]
	FMUL [sum]
	FABS
	FLD [prevTerm]
	FABS
	FCOMPP
	FSTSW ax
	SAHF
	jbe sumLoopEnd

	; Если же abs(prevTerm) > percentMultiplyer * sum, то
	; запускаем следующую итерацию цикла
	jmp sumLoop
sumLoopEnd:
	ret

endp

; Процедура вывода подсчитанного значения косинуса
proc printCosX

	invoke printf, cosXResultText, dword[x], dword[x + 4],\
				dword[sum], dword[sum + 4]
	add esp, 20

	ret
```

```
endp

; Процедура вывода "точного" значения косинуса с помощью FCOS
proc printRealCosX

        FLD [x]
        FCOS
        FSTP [floatTmp]
        invoke printf, realCosXResultText, dword[x], dword[x + 4],\
                                dword[floatTmp], dword[floatTmp + 4]
        add esp, 20

        ret

endp

; Секция импорта
section '.idata' import data readable

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

import kernel,\
       ExitProcess, 'ExitProcess'

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