Compute Cosine using Taylor Series

Overview

This project stresses the use of floating point instructions to create a program that computes the cosine of an angle given to you in degrees on the command line.

Taylor Series

The cosine of an angle given in radians can be found using the Taylor Series:

```
cosine x = 1 - x^2/2! + x^4/4! - x^6/6! ...
```

Notice each term flips from addition to subtraction.

Notice each term is based on the even integers starting at 2. The term, “1” is an exception since 02 is 0.

Command line

You are to accept two arguments on the command line. `getopt` is not being used here to concentrate on the floating point math. Both arguments are required.

- The angle in degrees whose cosine you wish to calculate. Take this to be a double.
- The number of terms to evaluate. The number of terms must lie between 1 and 10 inclusive.

C version

To assist your efforts, here is a version of this project written in C. This has been updated to print nice debugging output which is not part of the project. Only the last line is to be duplicated in your assembly language version.

This C version demonstrates one way of calculating the toggle. This version flips the sign of the toggle by multiplying by -1. When using assembly language, you can choose to emulate this but using `tbnz` is an alternative. You are free to use either method.

Sample executions

```
perrykivolowitz@Daedalus cos % !gc
gcc -o aversion main.S
perrykivolowitz@Daedalus cos % ./aversion 30 10
The cosine of 30.0000 degrees is 0.8660254038. Error: 0.0000000000
perrykivolowitz@Daedalus cos % ./aversion 45 10
The cosine of 45.0000 degrees is 0.7071067812. Error: 0.0000000000
perrykivolowitz@Daedalus cos % ./aversion 60 10
```

```
The cosine of 60.0000 degrees is 0.5000000000. Error: 0.0000000000
perrykivolowitz@Daedalus cos % ./aversion 90 10
The cosine of 90.0000 degrees is 0.0000000000. Error: -0.0000000000
perrykivolowitz@Daedalus cos % ./aversion 180 10
The cosine of 180.0000 degrees is -0.9999999999. Error: -0.0000000001
perrykivolowitz@Daedalus cos % ./aversion 360 10
The cosine of 360.0000 degrees is 1.0003012240. Error: -0.0003012240
perrykivolowitz@Daedalus cos %
```

Notice that as the angle increases away from 0, the accuracy of the approximation decreases. To maintain accuracy, more terms are needed. However, this quickly becomes impractical due to overflow or underflow.

The error term makes use of the C Runtime Library function `cos`.

The value you compute is subtracted from the value the CRT computes. The result is included in the print out.

Floating point instructions I used

These are the floating point instructions I used in my implementation.

- `fmov`
- `scvtf`
- `fmul`
- `fdiv`
- `fadd`
- `fsub`
- I also used `fneg` which you might not need.

How I broke up the program

I have functions named:

- main
- HandleOptions - gets, parses and checks the command line
- Factorial
- IntegerPower - x to the nth power
- ComputeSine - The main calculation
- PrintAnswer
- ConvertTheta - Wrap D2R
- D2R - Degrees to radians

Sad story

In writing the assembly language version of this project, I decided to check my intermediate answers with ChatGPT. It was easy enough to frame the request and I did so in many ways **getting a different wrong answer every time**. I wasted **hours** looking for a bug in my code ultimately to discover ChatGPT's lies and deceit.

For example, here are some gems:

Around three minus around five is about zero…

```
~ 3.14159 - (3.14159^3 / 3!)
~ 3.14159 - 4.9348
~ -0.0005
```

or this (ChatGPT even generated this code):

```c
# include <stdio.h>
# include <math.h>
int main() {
    const double pi = 3.14159265358979323846;
    double result = pi *(1 - pi*pi / 6.0);
    printf("%f\n", result);
    return 0;
}
```

This is a hard coded version of a 2 term Taylor Series evaluation of 180 degrees (PI radians).

And Chat's answer: 2.958040

The correct answer is: -2.026120.

Bottom line: ChatGPT lies as much as a politician.

ChatGPT for President!

CSC3510

The following applies to Carthage College CSC3510 students.

Work rules

Work is to be done solo.

What to hand in

Just the .S file. **Your name must be at the top of the file.**

Setting expectations

With extensive commenting, my solution is about 270 lines. This is not a challenge, rather a figure by which to set expectations. Should you find you're writing a thousand lines, for example, you're doing something wrong.