Department of Computer Science

COS132 2015
Practical 5

Mr Madoda Nxumalo

0 Introduction

0.1 Uploads and Deadline

This practical is an in-lab practical, and as such students must write it at the Informatorium during the normal practical session hours. The uploads will become available at practical session hours as of **17 April - 23 April**.

Make sure that you have submitted your code to Fitchfork all your tasks at the end of the practical session. **No late submissions will be accepted**.

0.2 Plagiarism Policy

All submissions for this practical should (as usual) be documented as specified in our plagiarism policy. Refer to the notes about our plagiarism policy which can be found on the COS132 website on ClickUP.

1 Trigonometry Series

The trigonometric functions are functions that are applied to an angle. The most common trigonometry functions are the Sine, Cosine and Tangent. The Sine and Cosine functions can be calculated using the following infinite series, where x is the angle in radians;

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, x \in \mathbb{R}$$
$$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, x \in \mathbb{R}$$

$x \in \mathbb{R}$ means that x is any real number. Note that the first terms of the functions can respectively be expressed as $\frac{x^1}{1!}$ and $\frac{x^0}{0!}$.

The tangent of an angle can be calculated using:
$tan(x) = \frac{sin(x)}{cos(x)}$, $cos(x) \neq 0$

$cos(x) \neq 0$ means "when $cos(x)$ is not zero" (when $cos(x)$ is equal to zero, the value of $tan(x)$ is undefined)

Implement the function defined in the following section.

2 Functions

`int` `factorial(`**`int`** `i)`

The factorial of n, denoted by $n!$ is the product $1 \times 2 \times 3 \times \cdots \times n$. The function should return the factorial of its parameter. $0!$ is defined to be the product of the natural numbers in the series with zero terms, i.e. the product of no values. Since the unit of multiplication is 1, this value is 1. If the parameter is negative, the function should return -1.

`float` `toRadians(`**`float`** `y)`

Converts an angle passed to it as parameter to the corresponding value in radians. The conversion formula is $rad(y) = y \times \frac{\pi}{180}$, where `y` is the angle in degrees.

`void` `sinSeries(`**`float`** `x,` **`double`** `v[],` **`size_t`** `t)`

fills the array passed as the parameter with the first t terms of the series for $sin(x)$ as given above, where x is the value passed in the first parameter.

`void` `cosSeries(`**`float`** `x,` **`double`** `v[],` **`size_t`** `t)`

fills the array passed as the parameter with the first t terms of the series for $cos(x)$ as given above, where x is the value passed in the first parameter.

`double` `sumOfSeries(`**`const double`** `v[],` **`size_t`** `t)`

returns the sum of the first t entries of the array passed as a parameter to the function.

void printSeries(**const double** v[], **size_t** t)

prints the series of which the first t terms are the entries in the array passed as a parameter in the following format:

$$v_0 + v_1 + v_2 + v_3 + \cdots + v_n.$$

void tanSeries(**float** x, **double** s[], **double** c[], **size_t** t)

fills the two arrays passed as the parameter with the first t terms of the series for $sin(x)$ and $cos(x)$ respectively, using the sinSeries and the cosSeries functions. x is the value passed in the first parameter.

3 The program

You are given an incomplete program that can be used to compare the values calculated using these series with values calculated using the functions in the **cmath** header.

Your program should make use of the above mentioned functions to produce output as shown in the following test runs:

3.1 Test run

```
Enter an angle in degrees:  30
1.  Calculate sin(30)
2.  Calculate cos(30)
3.  Calculate tan(30)
Enter your choice:  1
Enter the number of terms:  4
sin(30) = 0.523599 + -0.0239246 + 0.000327953 + -2.14072e-06 = 0.5
When using the cmath sin function:
sin(30) = 0.5
```

3.2 Test run

```
Enter an angle in degrees:  55
1.  Calculate sin(55)
2.  Calculate cos(55)
3.  Calculate tan(55)
Enter your choice:  2
Enter the number of terms:  5
cos(55) = 1 + -0.460734 + 0.0353793 + -0.0010867 + 1.78813e-05 = 0.573577
When using the cmath cos function:
cos(55) = 0.573576
```

3.3 Test run

```
Enter an angle in degrees:  60
1.  Calculate sin(60)
2.  Calculate cos(60)
3.  Calculate tan(60)
Enter your choice:  3
Enter the number of terms:  5
tan(60) =
1.0472 + -0.191397 + 0.0104945 + -0.000274012 + 4.17344e-06
------------------------------------------------------------
1 + -0.548311 + 0.0501076 + -0.00183164 + 3.58681e-05
= 1.73205
When using the cmath tan function:
tan(60) = 1.73205
```

4 Comments

- This given test suite does not have full coverage. You are encouraged to test your program with more test cases.
- **size_t** is an alias of one of the fundamental unsigned integer types. It is defined in `<cstddef>`. It is a type that is guaranteed to be able to represent any array index, or the size of any object in bytes. **size_t** is the type returned by the `sizeof` operator and is widely used in the standard library to represent sizes and counts. You should *prefer* using **`size_t`** instead of **`int`** or **`unsigned`** for variables that hold the size of an array, and for loop counters that represent array indexes. For example:

  ```
  for (size_t i = 0; i < len; i++)
      array[i] = ...;
  ```
- You should not use libraries other than those included in the given h-file.
- The accuracy of the values calculated using the series increases when more terms are used.

5 Submission

There are multiple upload slots for this session. Each upload slot focuses on testing only one of the functions. In each case you have to submit the following files:

- **main.cpp** - You are given an incomplete version of this program. You have to complete the program by replacing the comments with code.
- **series.h** - This file is given. It contains the prototypes of the functions. You should not change this file.
- **series.cpp** - You have to create this file. It should contain the implementations of the functions.
- **makefile** - you are provided with a makefile containing all the needed commands.

The tarball containing the above mentioned files can be created by the following command.

```
tar -cvz series.h series.cpp main.cpp makefile -f series.tar.gz
```

6 Marks Allocation

Task	**Percentage**	**Maximum mark**
factorial	6%	5
toRadian	6%	6
sinSeries	17%	12
cosSeries	17%	10
sumOfSeries	6%	10
printSeries	14%	6
tanSeries	18%	11
main	16%	9
Total	**100**	**69**