Tel-102 Paralelo 200
2º Semestre 2020
Control 1
29 de Septiembre de 2020

Instrucciones

- Lea con detenimiento y desarrolle **individualmente** cada una de las actividades a realizar durante la experiencia.
- Cree un archivo con extensión **.cpp** con lo desarrollado. El nombre del archivo debe tener el siguiente formato: **TEL102_C1_Nombre_Apellido.cpp** (Ej. TEL102_C1_Nicolas_Galvez.cpp), sin incluir tíldes.
- Enviar el archivo a través de la página de aula del ramo, sección "Control 1" hasta las 11:45:00 del día de **hoy**, Martes 29/09/2020, Hora continental de Chile (UTC-3).
- Cada minuto de atraso tendrá un descuento siguiendo la serie de Fibonacci.
- Trate de utilizar herramientas conocidas o aprendidas en clases. **No copie literalmente de recursos online**.
- Sea riguroso con las instrucciones de desarrollo.

Comunicaciones Saboteadas

Ud. está encandilado con **Between Us**, el juego de moda en **PlusPlus City**. En este, un grupo de 10 personajes se encuentran encerrados dentro de una nave espacial, y dos de ellos son asesinos que desean eliminar a resto de los tripulantes de la nave. Además, los asesinos tienen la capacidad de sabotear distintas funcionalidades de las nave espacial. Su ataque preferido es dañar la **Cabina de Comunicaciones**. Los tripulantes para arreglar ésta deben programar en C++ un pequeño código: Deben aproximar la función trigonométrica **coseno** con alta precisión.

El juego entrega el código, llamado `approx.cpp`, a los tripulantes para que puedan arreglar la cabina de comunicaciones en caso de sabotaje.

approx.cpp

```
#include<iostream>
#include<cmath>
#define N 100

struct valores{
   double aprox[N];
   double diff[N];
   int n;
};

// Acá se crean las funciones

// Segmento principal
int main(){
        // Programe aquí

        return 0;
}
```

Para poder reparar dicha facilidad ud. debe:

1. Crear una función `double cosAprox(double x, int n)`, que aproxima el cos del parámetro `double x` utilizando la siguiente serie matemática, y su aproximación hasta `int n`):

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

   Un ejemplo de la utilización de la función `cosAprox`, se puede inferir de la siguiente ejecución:

Salida (consola)

```
[foobar@control1 mauricio]$ ./approx
Ingrese valor: 0.75
Ingrese n: 10
Resultado: 0.731689
```

   Dada las restricciones de memoria y disco de la nave espacial, **se prohibe el uso de la biblioteca `<cmath>`** o similar.

2. Para analizar que los resultados de esta aproximación sean los suficientemente precisos, su programa de generar la aproximación necesaria incrementando el n hasta alcanzar **precisión** p determinada, esto es:

$$|\text{senoApprox(x,n)} - \textit{coseno}(\text{x})| \leq p$$

   Para calcular *coseno*(x) con el que va a comparar su aproximación, usted **debe utilizar la biblioteca** `<cmath>`, donde solo podrá ocupar la función `double cos(double x)` que viene con esa biblioteca.

   Un ejemplo de esto sería:

Salida (consola)

```
[foobar@control1 mauricio]$ ./approx
Ingrese valor: 0.75
Ingrese precisión: 0.0001
Resultado: 0.731686
Valor n: 4
```

3. Además, el mismo programa debe guardar cada una de las iteraciones de la aproximaciones del punto anterior (2) en una estructura de datos que contenga el cálculo de los cosenos, la diferencia absoluta entre su aproximación y el valor *"real"* del coseno, y el número de iteraciones n de la aproximación (véase el `struct` valores). Se establece que una aproximación que requiere más de 100 iteraciones ya es demasiado costosa, y por lo tanto no se continúa iterando.

   A partir de ésta estructura, se genera como resultado por pantalla, los valores de las aproximaciones de cosenos solo en aquellas iteraciones donde existió la mayor diferencia, la menor diferencia y el mejor resultado. Además imprima el número total de iteraciones de éste último. Un ejemplo de esto sería:

Salida (consola)

```
[foobar:control1 mauricio]$ ./approx
Ingrese valor: 0.75
Ingrese precisión: 0.0001
Valor máximo: 0.268311
Valor mínimo: 2.46751e-06
Mejor resultado: 0.731686
Iteraciones = 3
```