COMILLAS UNIVERSIDAD PONTIFICIA ICAI	Fundamentos de Informática 1º GITI	7/Mar/2023

Programa 1: Funciones para series (3 puntos)

Dada la siguiente serie:

$$1x + 3x/2! + 5x/3! + 7x/4! \ldots$$

Para calcular la aproximación de la serie para n términos se ha de utilizar la función `CalcularSerie()` con el siguiente prototipo:

```
float CalcularSerie(float x, int n);
```

Esta función recibe el valor de x y el número de términos, y devuelve la aproximación de la serie obtenida con ese número de términos.

La función `BuscarNumeroTerminos()` debe buscar el número de términos que da como resultado la mejor aproximación al valor `suma` sin sobrepasarlo. El prototipo de la función es:

```
int BuscarNumeroTerminos(float x, float suma);
```

Esta función utilizará a su vez la función `CalcularSerie()` para buscar el número de términos que obtiene la mejor aproximación.

Se pide:

- En el `main`, solicitar al usuario el valor de x y el valor de la serie, `suma`, al que se quiere aproximar (NO es necesario validar estos datos de entrada).
- Se llamará a la función `BuscarNumeroTerminos()` que buscará el número de términos más adecuado para conseguir la mejor aproximación al valor `suma` sin sobrepasarlo (mediante llamadas a la función `CalcularSerie()`) y devolverá al `main` ese número de términos.
- Finalmente, en el `main`, se mostrará el número de términos.

Es necesario programar **TODO** salvo la función `Factorial()`, que se puede usar, pero NO es necesario programar. La función `Factorial()` tiene el siguiente prototipo:

```
int Factorial(int num);
```

Nota: Se supone que siempre existe el número de términos buscado que se aproxima a la `suma`.

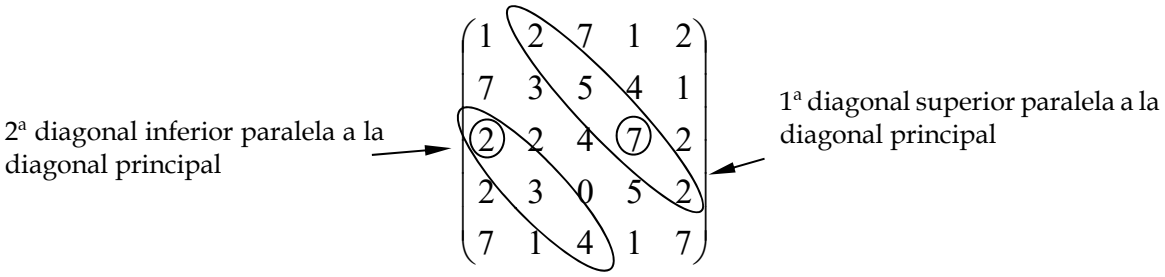
Ejemplo de ejecución:

```
Introduzca el valor al que desea aproximar la serie: 3.7
Introduzca el valor de x: 1

La serie de 5 terminos es la que mejor aproximacion obtiene al valor 3.70
Time elapsed: 000:05:281
Press any key to continue
```

Programa 2: Máximos y mínimos en diagonal (3,5 puntos)

Un profesor del Departamento de Telemática y Computación de ICAI, está realizando un estudio sobre las matrices cuadradas de números de dimensión entre 2 y 5. En particular, se desea conocer para cada diagonal superior paralela a la diagonal principal el máximo valor de sus elementos y, para cada diagonal inferior paralela a la diagonal principal, el mínimo valor de sus elementos. Por ejemplo, en la matriz de la figura, la primera diagonal superior tiene el valor máximo 7 y la segunda diagonal inferior tiene el valor mínimo 2.



Se pide diseñar un programa completo en C que realice secuencialmente las siguientes tareas:

a) Pedir la dimensión `dim` de la matriz, que debe ser como máximo 5 y como mínimo 2 (se debe validar la entrada).
b) Posteriormente, se pedirá al usuario cada uno de los valores de la matriz. Dicha tarea se realizará implementando la función:

```
void Leer_Matriz(int datos[][5], int dim);
```

c) A continuación, se deben pedir al usuario dos datos: una variable **`sup_inf`** de tipo **`char`** y otra variable **`diag`** de tipo **`int`** (NO es necesario validar estos datos de entrada). Estas variables determinarán el valor que se ha de buscar en la matriz. Así, si la variable `sup_inf` es igual a `'S'` o `'s'` significará que queremos buscar en una diagonal superior o por encima de la diagonal principal. Si por el contrario `sup_inf` es igual a `'I'` o `'i'` significará que queremos buscar en una diagonal inferior o por debajo de la diagonal principal. La variable `diag` indicará a continuación el número de diagonal superior o inferior dónde se desea buscar.

   - Por ejemplo, si `sup_inf='S'` y `diag=1`, significa que debemos buscar el máximo de la primera diagonal superior (en el ejemplo del gráfico el número 7).
   - De otra forma si por ejemplo `sup_inf='I'` y `diag=2` significa que debemos buscar el valor más pequeño de la segunda diagonal inferior (en el ejemplo del gráfico el número 2).

   Una vez leídas las variables `sup_inf` y `diag` que identifican el elemento a buscar, se debe llamar a la función `Cacula_Elemento()` para buscar el elemento correspondiente. El prototipo de esta función (que se debe implementar) es:

```
int Calcula_Elemento (int datos[][5], int dim, char sup_inf, int diag);
```

d) Para finalizar el programa, en `main` se imprimirá por pantalla la matriz original datos, las variables `sup_inf` y `diag` y el valor obtenido.

Programa 3: Estimación de PI mediante Montecarlo (3,5 puntos)

En matemáticas, Montecarlo es un método usado para aproximar con exactitud expresiones matemáticas complejas de evaluar. Utilizando este método, cuyos pasos se indican posteriormente, se pide desarrollar un programa completo en C que realice 10 simulaciones distintas para estimar el número PI (π). Realizadas las 10 simulaciones, se mostrará en cuál de ellas se ha obtenido el valor más aproximado a PI y el error cometido, debiendo quedar los datos empleados en dicha simulación almacenados en un vector.

Para aplicar Montecarlo en la estimación de PI, se deben ir generando de forma aleatoria valores de *x* (abcisa) e *y* (ordenada) en el intervalo [0,1], hasta obtener N puntos (x,y). A medida que se generan los puntos, se irán contabilizando aquellos que cumplen con la Expresión [1]:

Expresión [1] $$x^2 + y^2 \leq 1$$

El valor aproximado de PI se obtendrá mediante la siguiente fórmula:

$$\pi = \frac{4 \cdot C}{N}$$

donde N es el número total de puntos generados para una simulación, mientras que C es la cantidad de puntos, de los N generados, que cumplen con la Expresión [1].

Notas

- La función rand() devuelve un entero entre 0 y RAND_MAX (constante de tipo `int` ya definida en C). El número devuelto se dividirá entre RAND_MAX para obtener un número en el intervalo [0,1]. Mediante `srand((unsigned)time(NULL));` se genera la semilla para la función `rand()`. Para el uso de `rand()`, `srand()` y `time()`, se precisan las librerías `stdlib.h` y `time.h`.
- Para calcular el error en la estimación de PI (la diferencia entre el valor real de PI y el valor de PI estimado) se utilizará `M_PI`, constante ya definida en C (librería `math.h`) con el valor real de PI.

Programa

- Definir la constante simbólica N para establecer en 10.000 los puntos (x,y) que se generarán para cada simulación.
- Declarar dos vectores en `main` (`vec_curso` y `vec_optimo`) ambos de tipo `double` y tamaño 2*N. Para cada simulación, los puntos (x,y) generados se irán guardando en `vec_curso`. Si el error obtenido en una simulación es el menor de los errores de las simulaciones realizadas hasta el momento, se copiará el contenido completo de `vec_curso` en `vec_optimo`. La disposición de las coordenadas de cada punto en los vectores, comenzando por (x_0,y_0), es la siguiente:

	X_0	Y_0	X_1	Y_1	…	X_{N-1}	Y_{N-1}
Índices del vector:	*0*	*1*	*2*	*3*		*2*N-2*	*2*N-1*

- Para cada una de las 10 simulaciones, desde `main` se llamará a la función `EstimarPI()`, que generará los N puntos aleatorios para la simulación en curso, irá guardando las coordenadas en `vec_curso`, calculará PI estimado y devolverá el error, en valor absoluto, de la estimación de PI. El prototipo de la función `EstimarPI()` es el siguiente:

```
double EstimarPI(double vector[]);
```

- Por último, el programa indicará por pantalla en cuál de las 10 simulaciones se ha obtenido el valor más aproximado a PI y mostrará el valor del error de esa simulación.

Ejemplo de ejecución:

```
La Simulación 5 ha sido la óptima. Error = 0.0000073464
```