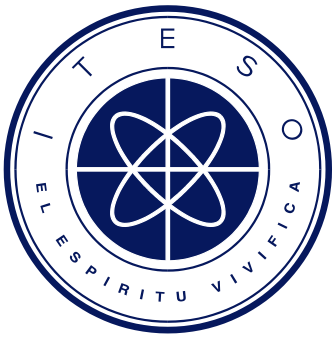
Instituto Tecnológico y de Estudios Superiores de Occidente

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. APUNTADORES A FUNCIONES

Autor: Medina Rifas Edgar Francisco

Presentación: 5 pts.
Funcionalidad: 60 pts.
Pruebas: 10 pts.

4 de junio de 2018. Tlaquepaque, Jalisco,

- Las figuras deben tener un número y descripción.
- Las figuras, tablas, diagramas y algoritmos en un documento, son material de apoyo para transmitir ideas.
- Sin embargo deben estar descritas en el texto y hacer referencia a ellas. Por ejemplo: En la Figura 1….
- Falta describir las pruebas (escenario, y resultados de la experimentación).
- Cuando se tienen resultados que se pueden comparar, se recomienda hacer uso de diagramas o tablas que permitan observar el resultado de los diversos casos y contrastas los resultados (en el tiempo por ejemplo).

Instrucciones para entrega de tarea

Esta tarea, como el resto, es ***IMPRESCINDIBLE*** entregar los entregables de esta actividad de la siguiente manera:

- **Reporte:** vía *moodle* en **un archivo PDF**.
- **Código:** vía su repositorio **Github**.

La evaluación de la tarea comprende:

- 10% para la presentación
- 60% para la funcionalidad
- 30% para las pruebas

Es necesario responder el apartado de conclusiones, pero no se trata de llenarlo con paja. Si no se aprendió nada al hacer la práctica, es preferible escribir eso. Si el apartado queda vacío, se restarán puntos al porcentaje de presentación.

Objetivo de la actividad

El objetivo de la tarea es que el alumno aplique los conocimientos y habilidades adquiridos en el tema de apuntadores a funciones y la distribución de tareas mediante el uso de hilos para la resolución de problemas utilizando el lenguaje ANSI C.

Descripción del problema

Existen diversas técnicas para generar una aproximación del valor del número irracional **Pi**. En este caso utilizaremos la serie de Gregory y Leibniz.

$$\pi = 4\left(\sum_{n=1}^{\infty}\left(\frac{(-1)^{(n+1)}}{(2n-1)}\right)\right)$$

$$= \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right)$$

Procedimiento

1. Codificar una solución secuencial (sin el uso de hilos) que calcule el valor de Pi, su solución debe basarse en la serie de Gregory y Leibniz para calcular los primeros diez dígitos decimales de Pi. Para esto, utilice los primeros tomando los primeros 50,000’000,000 términos de la seria.

2. Utilice las funciones definidas en la librería **time.h** (consulte diapositivas del curso) para medir el tiempo (en milisegundos) que requiere el cálculo del valor de **Pi**. Registre el tiempo.

3. Parametrice la solución que se implementó en el paso 1.

4. Utilice hilos para repartir el trabajo de calcular el valor de **Pi**. Pruebe su solución con los siguientes casos: 2 hilos, 4 hilos, 8 hilos y 16 hilos.

5. Tomar el tiempo en milisegundos que toma el programa para calcular el valor de **Pi** en cada uno de los casos mencionados en el paso 4.
6. Registre los tiempos registrados para cada caso en la siguiente tabla:

No. de Hilos	Tiempo (milisegundos)
1	782459
2	417318
4	212046
8	114651
16	115123

Descripción de la entrada

El usuario deberá indicar al programa cuantos hilos quiere utilizar para el calcular el valor de **Pi**.

Descripción de la salida

En un renglón imprimirá el valor calculado de **Pi**, con exactamente 10 dígitos decimales. En el siguiente renglón mostrará el número de milisegundos que se requirió para realizar el cálculo.

Ejemplo de ejecución:

```
¿Hilos? 4
Pi:     3.1415926535
Tiempo: 24487 ms
```

SOLUCIÓN DEL ALUMNO, PRUEBAS Y CONCLUSIONES

Código fuente de la versión secuencial (sin el uso de hilos)

```
/*IS715468
 * main.c
 *
 *  Created on: 05/06/2018
 *      Author: Edgar Francisco Medina Rifas
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
double Pi (long long int fin);

int main()
{

      setvbuf(stderr, NULL, _IONBF, 0);
      setvbuf(stdout, NULL, _IONBF, 0);
      double acum;
      clock_t start=clock();
      acum=Pi(50000000000);
      clock_t stop= clock();
      int ms = 1000 * (stop-start)/CLOCKS_PER_SEC;
      printf("PI: %.10lf\n",acum);
      printf("Tiempo: %d",ms);
      return 0;
}
```

```
double Pi (long long int fin)
{
	double suma=0;
	long long int i;
	for (i=1;i<=fin; i++)
	{
		suma+=((i+1) & 1 ? -1.0: 1.0)/(2*i-1);
	}
	return 4*suma;
}
```

Código fuente de la versión paralelizada

```
/*IS715468
 * main.c
 *
 *  Created on: 05/06/2018
 *      Author: Edgar Francisco Medina Rifas.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include <inttypes.h>
//Variable global para la aproximación de PI.
double suma=0;
DWORD WINAPI piA(void *);

typedef struct
{
	long long int limiteI;
	long long int limiteM;
}param;

int main()
{
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	//Arreglo de Handles
	HANDLE h[16];
	int hilos;
	//Arreglo de parametros para los futuros hilos.
	param datos[16];
	long long int razon=50000000000;
```

```
	printf("Cuantos hilos quieres: ");
	scanf("%d",&hilos);
	razon=razon/hilos;
	//Comienza el cronometro de milisegundos.
	clock_t start=clock();
	//Creación de hilos y sus respectivos rangos. Mediante
formula de diferenciales en calculo.
	for (int i=0;i<hilos;i++)
	{
		datos[i].limiteI=(razon*i)+1;
		datos[i].limiteM=razon*(i+1);
		h[i]=CreateThread(NULL,0,piA,(void *)&datos[i],0,NULL);
	}
	//For para esperar que todos los hilos terminen de correr.
	for (int i=0; i<hilos;i++)
	{
		WaitForSingleObject(h[i],INFINITE);
	}
	clock_t stop= clock();
	int ms = 1000 * (stop-start)/CLOCKS_PER_SEC;
	printf("PI: %.10lf\n",suma);
	printf("Tiempo: %d ms\n",ms);
	system("Pause");
	return 0;
}
//Función para hilos que aproxima el valor a Pi con
50,000,000,000.
DWORD WINAPI piA(void * arg)
{
	param *arreglo= (param *) arg;
	double temp=0;
	long long int i;
	for (i=arreglo->limiteI;i<=arreglo->limiteM; i++)
	{
		temp+=((i + 1) & 1 ? -1.0 : 1.0)/(2*i-1);
	}
	temp*=4;
	suma+=temp;
	return 0;
}
```

Ejecución

```
Cuantos hilos quieres: 16
PI: 3.1415926536
Tiempo: 115123 ms
Presione una tecla para continuar . . .
```

```
PI: 3.1415926536
Tiempo: 804710
```

Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
  - Aprendí a crear hilos. No tenía idea de como funcionaban, pero después de esta practica ya creo saber más sobre ellos. Me gustó ver la diferencia entre los tiempos al tener diferentes hilos. Entendí los parámetros y aparte implemente "cast" (esto ya me quedó más claro).
- ✓ Lo que me costó trabajo y cómo lo solucioné.
  - La creación e implementación de los hilos. Ajustar los tipos de datos para poder realizar las operaciones esperadas. Reajustar el iterador de -1 y 1 ya que la función pow hacía que el tiempo en ejecución se ralentizará.
- ✓ Lo que no pude solucionar.