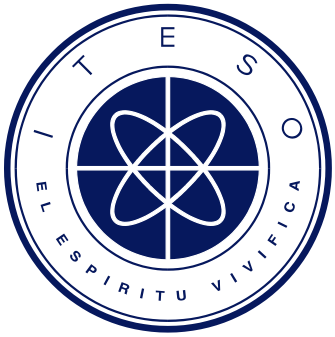
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. APUNTADORES A FUNCIONES

Autor: Soto Pérez, Omar

Presentación: 5 pts.
Funcionalidad: 60 pts.
Pruebas: 20 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).

Compara tu implementación con la de otros compañeros.

Instrucciónes 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 suo 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 implemento 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	1037453 ms
2	531705 ms
4	285402 ms
8	285438 ms
16	286789 ms

Descripción de la entrada

El usuario eberá 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)

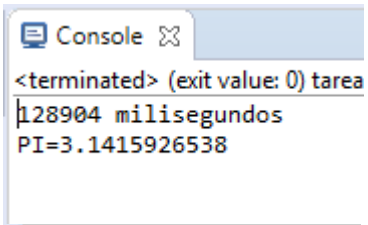
```
#include<stdio.h>
#include<stdlib.h>
#include<windows.h>
#include<time.h>

int main()
{
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);

	double pi,aux;
	double acum=0;
	long long int n=50000000000;
	clock_t start=clock();
	for(long long int i=1;i<n;i++)
	{
		aux=((i+1)&1?-1.0:1)/(2*i-1);
		acum+=aux;
	}
	pi=acum*4;
	clock_t stop=clock();
	long int mil=1000*(stop-start)/CLOCKS_PER_SEC;
	printf("%li milisegundos\n",mil);
	printf("PI=%.10f\n",pi);

	return 0;

}
```

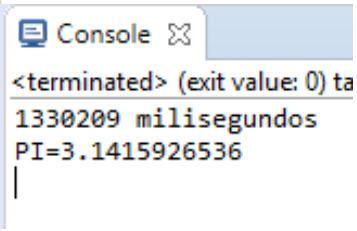
Código fuente de la versión paralelizada

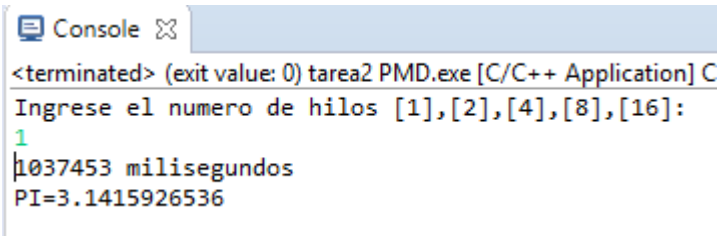
```
#include<stdio.h>
#include<stdlib.h>
#include<windows.h>
#include<time.h>

typedef struct
{
	long long int limS;
	long long int limI;
}limites;

double pifinal=0;

DWORD WINAPI function(void* lim);

int main()
{
```

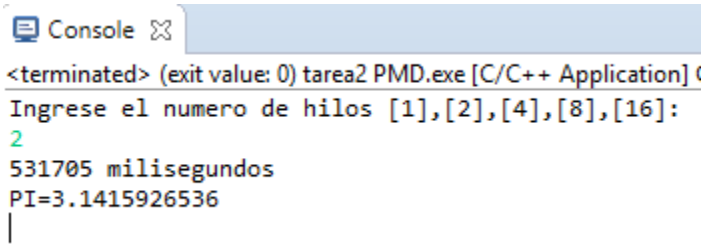
```
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);

	HANDLE H[16];
	int h;
	limites lim[16];
	long long int n=50000000000;
	printf("Ingrese el numero de hilos [1],[2],[4],[8],[16]:\n");
	scanf("%d",&h);
	n/=h;
	clock_t start=clock();
	for(long long int i=0;i<h;i++)
	{
		lim[i].limI=(n*i)+1;
		lim[i].limS=n*(i+1);
		H[i]=CreateThread(NULL,0,function,(void*)&lim[i],0,NULL);
	}
	for(long long int i=0;i<h;i++)
		WaitForSingleObject(H[i],INFINITE);

	clock_t stop=clock();
	long int mil=1000*(stop-start)/CLOCKS_PER_SEC;
	printf("%li milisegundos\n",mil);
	printf("PI=%.10f\n",pifinal);

//	for(long long int i=1;i<n;i++)
//	{
//		aux=((i+1)&1?-1.0:1)/(2*i-1);
//		acum+=aux;
//	}
//	pi=acum*4;



	return 0;
}
DWORD WINAPI function(void* lim)
{
	limites *auxiliar=(limites*)lim;
	double aux=0;
	for(long long int i=(*auxiliar).limI;i<=(*auxiliar).limS;i++)
		aux+=((i+1)&1?-1.0:1)/(2*i-1);
	aux*=4;
	pifinal+=aux;
	return 0;

}
```

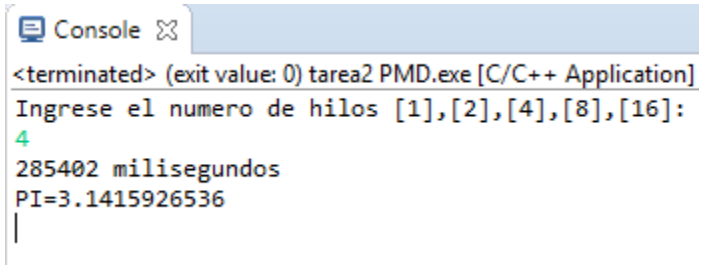
Ejecución

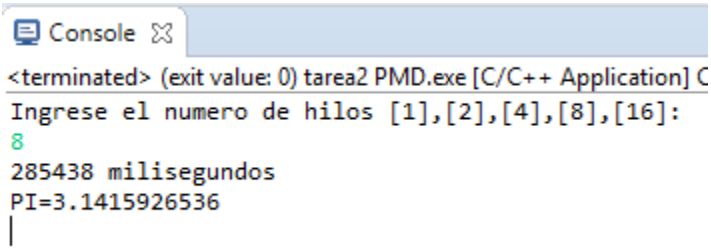

Console
<terminated> (exit value: 0) tarea
128904 milisegundos
PI=3.1415926538

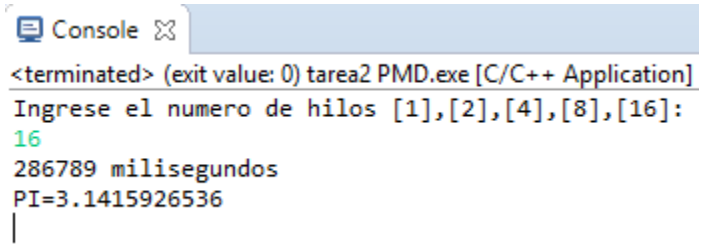

Console
<terminated> (exit value: 0) ta
1330209 milisegundos
PI=3.1415926536


Console
<terminated> (exit value: 0) tarea2 PMD.exe [C/C++ Application] C
Ingrese el numero de hilos [1],[2],[4],[8],[16]:
1
1037453 milisegundos
PI=3.1415926536


Console
<terminated> (exit value: 0) tarea2 PMD.exe [C/C++ Application]
Ingrese el numero de hilos [1],[2],[4],[8],[16]:
2
531705 milisegundos
PI=3.1415926536


Console
<terminated> (exit value: 0) tarea2 PMD.exe [C/C++ Application]
Ingrese el numero de hilos [1],[2],[4],[8],[16]:
4
285402 milisegundos
PI=3.1415926536


Console
<terminated> (exit value: 0) tarea2 PMD.exe [C/C++ Application] C
Ingrese el numero de hilos [1],[2],[4],[8],[16]:
8
285438 milisegundos
PI=3.1415926536


Console
<terminated> (exit value: 0) tarea2 PMD.exe [C/C++ Application]
Ingrese el numero de hilos [1],[2],[4],[8],[16]:
16
286789 milisegundos
PI=3.1415926536

Conclusiones (obligatorio):

Esta práctica me ayudo principalmente a entender el uso de los hilos y la funcionalidad que pueden llegar a tener, pues realmente era un tema del que nunca había escuchado y utilizado. En sí, lo que me costó más trabajo, fue entender la sintaxis de estos, pues no son funciones que haya utilizado en el pasado. En esta tarea puede solucionar todos los requerimientos, sin embargo, surgió una irregularidad, pues al momento de utilizar 16 hilos y el tiempo salía superior a cuando usaba 8 hilos.