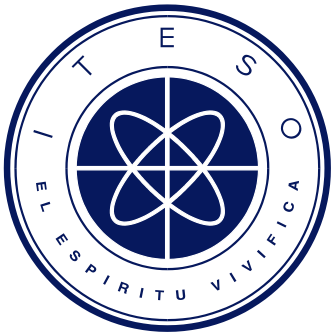
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. APUNTADORES A FUNCIONES

Autor: Salazar Valdovinos, Sofía

Presentación: 10 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).

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	2609
2	2873
4	2867
8	3255
16	2506

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)

```
/*
 ============================================================================
 Name        : Tarea2.c
 Author      : Momo
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
double Pi(long long limit);
int main(void) {
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	clock_t inicio, fin;

	double segundos;
	double miPi = 0;
	long long i, limit = 500000000;
	printf("Comenzando\n");
	inicio = clock();


	for(i=1; i <= limit ; i++){
		if((i&1)==1){
			//Imar
			miPi += (double)1/(2*i-1);
		}else{
			//Par
			miPi -= (double)1/(2*i-1);
		}
	}
	miPi = 4*miPi;
	fin = clock();
	printf("Valor de Pi: %.10g\n", miPi);
	segundos = (double)(fin - inicio) / CLOCKS_PER_SEC;
	printf("Tiempo requerido: %g milisegundos", segundos*1000.0);
	return EXIT_SUCCESS;
}
/*double Pi(long long limit){
	double pi = 0;
	long long  i;
	for(i=1; i < limit ; i++){
		if((i&1)==1){
			//Imar
			pi += (double)1/(2*i-1);
```

```
		}else{
			//Par
			pi -= (double)1/(2*i-1);
		}
	}
	return 4*pi;
}*/
```

Código fuente de la versión paralelizada

```
/*
 ============================================================================
 Name        : Tarea2_hilos.c
 Author      : Momo
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================================
 */

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


typedef struct{
	unsigned int inicio;
	unsigned int fin;
	double sumaHilo;
}Hilo;
double sumaTotal(void *param, int cant_hilos);
DWORD WINAPI suma(void *param);
int main(void) {
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	clock_t inicio, fin;
	int hilos;
	double segundos=0, pi=0;
	printf("Hilos [1,2,4,8,16]?");
	scanf("%d",&hilos);
	while(hilos==1 || hilos== 2 || hilos== 4 || hilos== 8 ||hilos==16){


		if(hilos == 1){
			Hilo hilo={1,500000000,0};
			HANDLE h1 = NULL;
			inicio = clock();
			h1 = CreateThread(NULL,0,suma,(void*)&hilo,0,NULL);
			WaitForSingleObject(h1, INFINITE);
			pi= sumaTotal(&hilo,1);
			fin = clock();
```

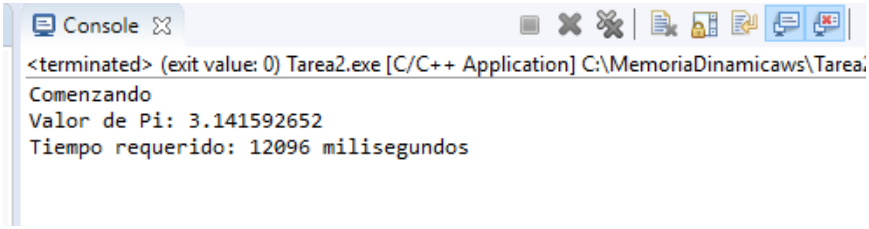
```
		}
		if(hilos == 2){
			Hilo hilo[2]={{1,250000000,0},{250000001,500000000,0}};
			HANDLE h1=NULL, h2=NULL;
			inicio = clock();
			h1=CreateThread(NULL, 0, suma, (void *)&hilo[0], 0, NULL);
			h2=CreateThread(NULL, 0, suma, (void *)&hilo[1], 0, NULL);
			WaitForSingleObject(h1, INFINITE);
			WaitForSingleObject(h2, INFINITE);
			pi= sumaTotal(hilo,2);
			fin = clock();
		}
		if(hilos == 4){
			Hilo
hilo[4]={{1,125000000,0},{125000001,250000000,0},{250000001,275000000,0},{275000
001,500000000}};
			HANDLE h1=NULL, h2=NULL,h3=NULL, h4 = NULL;
			inicio = clock();
			h1=CreateThread(NULL, 0, suma, (void *)&hilo[0], 0, NULL);
			h2=CreateThread(NULL, 0, suma, (void *)&hilo[1], 0, NULL);
			h3=CreateThread(NULL, 0, suma, (void *)&hilo[2], 0, NULL);
			h4=CreateThread(NULL, 0, suma, (void *)&hilo[3], 0, NULL);
			WaitForSingleObject(h1, INFINITE);
			WaitForSingleObject(h2, INFINITE);
			WaitForSingleObject(h3, INFINITE);
			WaitForSingleObject(h4, INFINITE);
			pi= sumaTotal(hilo,4);
			fin = clock();

		}
		if(hilos == 8){
			Hilo hilo[8]={

	{1,100000000,0},{100000001,150000000,0},{150000001,200000000,0},{20000000
1,250000000,0},

	{250000001,300000000,0},{300000001,380000000,0},{380000001,400000000,0},{
400000001,500000000,0}
			};
			HANDLE h1=NULL, h2=NULL,h3=NULL, h4 = NULL, h5= NULL, h6 =
NULL, h7=NULL, h8= NULL;
			inicio = clock();
			h1=CreateThread(NULL, 0, suma, (void *)&hilo[0], 0, NULL);
			h2=CreateThread(NULL, 0, suma, (void *)&hilo[1], 0, NULL);
			h3=CreateThread(NULL, 0, suma, (void *)&hilo[2], 0, NULL);
			h4=CreateThread(NULL, 0, suma, (void *)&hilo[3], 0, NULL);
			h5=CreateThread(NULL, 0, suma, (void *)&hilo[4], 0, NULL);
			h6=CreateThread(NULL, 0, suma, (void *)&hilo[5], 0, NULL);
			h7=CreateThread(NULL, 0, suma, (void *)&hilo[6], 0, NULL);
			h8=CreateThread(NULL, 0, suma, (void *)&hilo[7], 0, NULL);
			WaitForSingleObject(h1, INFINITE);
			WaitForSingleObject(h2, INFINITE);
			WaitForSingleObject(h3, INFINITE);
			WaitForSingleObject(h4, INFINITE);
			WaitForSingleObject(h5, INFINITE);
```

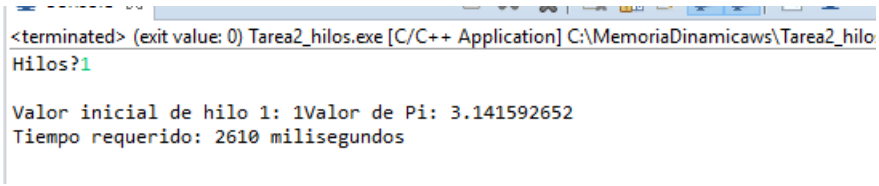
```
                        WaitForSingleObject(h6, INFINITE);
                        WaitForSingleObject(h7, INFINITE);
                        WaitForSingleObject(h8, INFINITE);
                        pi= sumaTotal(hilo,8);
                        fin = clock();
                }
                if(hilos == 16){
                        Hilo hilo[16]={

        {1,50000000,0},{50000001,80000000,0},{80000001,100000000,0},{100000001,12
5000000,0},

        {125000001,180000000,0},{180000001,200000000,0},{200000001,230000000,0},{
230000001,250000000,0},

        {250000001,290000000,0},{290000001,330000000,0},{330000001,355000000,0},{
355000001,375000000,0},

        {375000001,400000000,0},{400000001,440000000,0},{440000001,475000000,0},{
475000001,500000000,0}
                        };
                        HANDLE h1=NULL, h2=NULL,h3=NULL, h4 = NULL, h5= NULL, h6 =
NULL, h7=NULL, h8= NULL,h9= NULL, h10= NULL, h11= NULL, h12 = NULL,h13 =
NULL,h14 = NULL,h15 = NULL,h16= NULL;
                        inicio = clock();
                        h1=CreateThread(NULL, 0, suma, (void *)&hilo[0], 0, NULL);
                        h2=CreateThread(NULL, 0, suma, (void *)&hilo[1], 0, NULL);
                        h3=CreateThread(NULL, 0, suma, (void *)&hilo[2], 0, NULL);
                        h4=CreateThread(NULL, 0, suma, (void *)&hilo[3], 0, NULL);
                        h5=CreateThread(NULL, 0, suma, (void *)&hilo[4], 0, NULL);
                        h6=CreateThread(NULL, 0, suma, (void *)&hilo[5], 0, NULL);
                        h7=CreateThread(NULL, 0, suma, (void *)&hilo[6], 0, NULL);
                        h8=CreateThread(NULL, 0, suma, (void *)&hilo[7], 0, NULL);
                        h9=CreateThread(NULL, 0, suma, (void *)&hilo[8], 0, NULL);
                        h10=CreateThread(NULL, 0, suma, (void *)&hilo[9], 0, NULL);
                        h11=CreateThread(NULL, 0, suma, (void *)&hilo[10], 0, NULL);
                        h12=CreateThread(NULL, 0, suma, (void *)&hilo[11], 0, NULL);
                        h13=CreateThread(NULL, 0, suma, (void *)&hilo[12], 0, NULL);
                        h14=CreateThread(NULL, 0, suma, (void *)&hilo[13], 0, NULL);
                        h15=CreateThread(NULL, 0, suma, (void *)&hilo[14], 0, NULL);
                        h16=CreateThread(NULL, 0, suma, (void *)&hilo[15], 0, NULL);
                        WaitForSingleObject(h1, INFINITE);
                        WaitForSingleObject(h2, INFINITE);
                        WaitForSingleObject(h3, INFINITE);
                        WaitForSingleObject(h4, INFINITE);
                        WaitForSingleObject(h5, INFINITE);
                        WaitForSingleObject(h6, INFINITE);
                        WaitForSingleObject(h7, INFINITE);
                        WaitForSingleObject(h8, INFINITE);
                        WaitForSingleObject(h9, INFINITE);
                        WaitForSingleObject(h10, INFINITE);
                        WaitForSingleObject(h11, INFINITE);
                        WaitForSingleObject(h12, INFINITE);
                        WaitForSingleObject(h13, INFINITE);
                        WaitForSingleObject(h14, INFINITE);
```

```
                        WaitForSingleObject(h15, INFINITE);
                        WaitForSingleObject(h16, INFINITE);
                        pi= sumaTotal(hilo,16);
                        fin = clock();
                }
                segundos = (double)(fin - inicio) / CLOCKS_PER_SEC;
                printf("\nValor de Pi: %.10g\n",pi);
                printf("Tiempo requerido: %g milisegundos\n\n", segundos*1000.0);
                segundos = 0;
                inicio = 0;
                fin = 0;
                pi = 0;
                hilos = 0;
                printf("Hilos [1,2,4,8,16]?");
                scanf("%d",&hilos);
        }
        return EXIT_SUCCESS;
}

DWORD WINAPI suma(void *param){
        Hilo *p = param;
        long i;
        for(i = p->inicio; i <= (p->fin) ; i++){
                if((i&1)==1){
                        //Impar
                        p->sumaHilo += (double)1/(2*i-1);
                }else{
                        //Par
                        p->sumaHilo -= (double)1/(2*i-1);
                }
        }
        return 0;
}
double sumaTotal(void *param, int cant_hilos){
        Hilo *p = (Hilo*) param;
        double pi = 0;
        int i;
        for (i=0; i < cant_hilos; i++){
                //printf("\nValor inicial de hilo %d: %u", i+1, (p+i)->inicio);
                pi += (p+i)->sumaHilo;
        }
        return 4*pi;
}
```

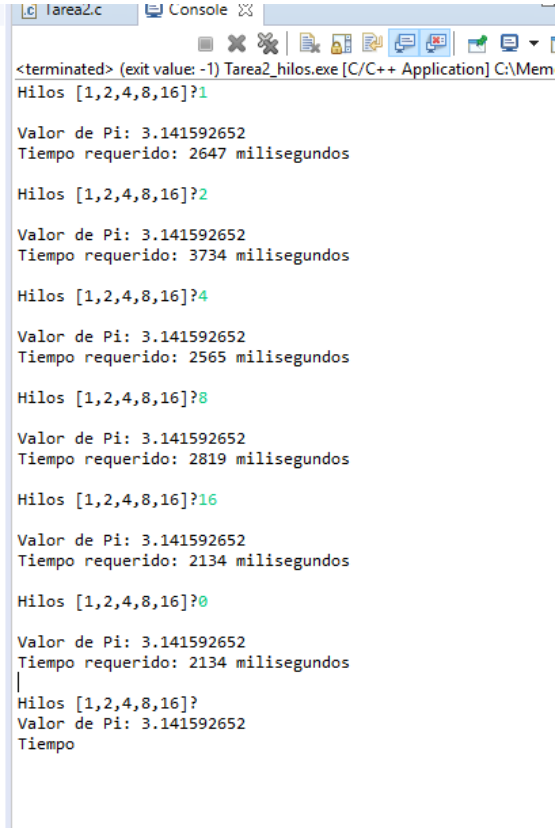
Ejecución

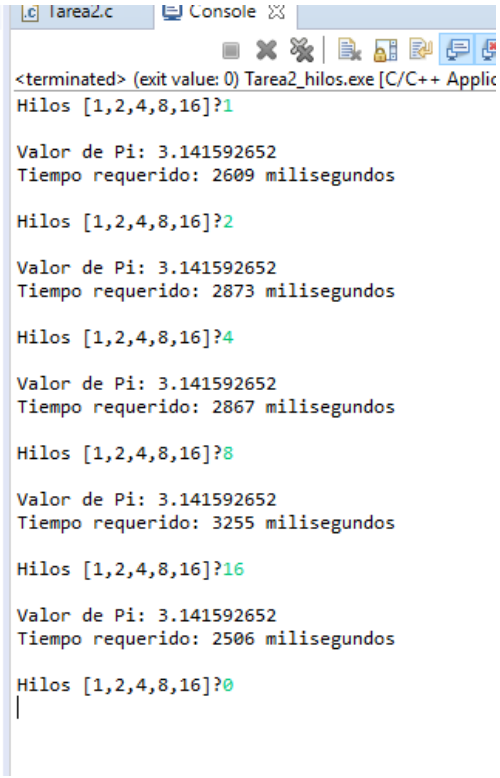
Ejecución secuencial con limite de 500’000,000:





Más Hilos:





Conclusiones (obligatorio):

- ✓ En esta práctica reforcé el uso de hilos y su implementación con funciones. Lo que aún me falta por saber es utilizar bien los parámetros de la función `CreateThread`. Otra de las cosas que reforcé fue el uso de apuntadores hacia estructuras. Además, el uso de la función clock fue algo nuevo para mí. Al final, pude apreciar el tiempo de los hilos pero solamente en un límite de 500 millones y no 50mil.