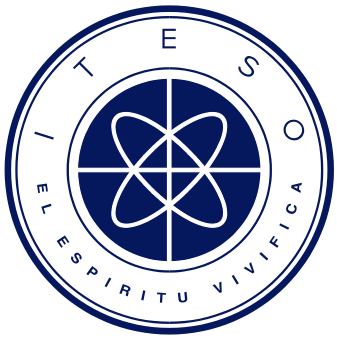
# INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

---


Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES


# PROGRAMACIÓN CON MEMORIA DINÁMICA

## TAREA 2. APUNTADORES A FUNCIONES


Autor: Méndez Méndez Sergio Alejandro


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).

## 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 | 568718 |
| 2 | 485053 |
| 4 | 488128 |
| 8 | 487099 |
| 16 | 487278 |

## SOLUCIÓN DEL ALUMNO, PRUEBAS Y CONCLUSIONES

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

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

double calcularPi(){
	double sum = 0;
	unsigned long denom = 1;
	int signo = 1;
	for (long i=1; i<=3125000000; i++){
		sum += signo*4.0/denom;
		denom += 2;
		signo *= -1;
	}
	return sum;
}

int main(void) {
	clock_t tiempo1, tiempo2, tfinal;
	int num;
	printf("Numero de hilos: ");
	scanf("%d", &num);
	tiempo1=clock();
	printf("Pi:\t%.10lf\n", calcularPi());
	tiempo2=clock();
	tfinal = (double)(tiempo2-tiempo1)/1000;
	printf("Tiempo:\t%lu ms", tfinal);
```

## Código fuente de la versión paralelizada

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

typedef struct{
	unsigned long long inicio;
	unsigned long long final;
	long double sumaParcial;
}Rango;

void * suma(void *parametro);


int main(void) {
	clock_t tiempo1, tiempo2;
	long tfinal;
	int num, i;
	long double sumaPi = 0;
	printf("Numero de hilos: ");
	scanf("%d", &num);

	Rango rango[num];
	rango[0].inicio = 1;
	rango[0].final = 50000000000/num;
	rango[0].sumaParcial = 0;
	for(i=1; i<num; i++){
		rango[i].sumaParcial = 0;
		rango[i].inicio = rango[i-1].final + 1;
		rango[i].final = rango[i].inicio + 50000000000/num;
	}
	rango[num-1].final = 50000000000;

	tiempo1=clock();

	pthread_t thread[num];
	for(i=0; i<num; i++){
		pthread_create(&thread[i], NULL, suma, (void *)
&rango[i]);
	}

	for(i=0; i<num; i++){
```

```
            pthread_join(thread[i], NULL);
        }

        tiempo2=clock();
        tfinal = (tiempo2-tiempo1)/1000;

        for(i=0; i<num; i++){
            sumaPi += rango[i].sumaParcial;
        }
        printf("%.10LF\n", sumaPi);
        printf("Tiempo:\t%lu ms", tfinal);

        return EXIT_SUCCESS;
}

void * suma(void *param){
        Rango *r = (Rango*)param;
        int signo;
        long long i;
        long double suma = 0;
        long long denom;

        if (r->inicio%2 == 0){
            signo = -1;
        } else {
            signo = 1;
        }

        denom = 2*r->inicio - 1;
        for(i=r->inicio; i<=r->final;i++){
            suma += signo*4.0/(denom);
            signo *= -1;
            denom += 2;
        }
        r->sumaParcial = suma;
        return NULL;
}
```

## Ejecución

### Secuencial

```
Numero de hilos: 1
Pi:     3.1415926536
Tiempo: 568718 ms
```

### Con Hilos

```
Numero de hilos: 2
3.1415926536
Tiempo: 485053 ms
```

```
Numero de hilos: 4
3.1415926536
Tiempo: 488128 ms
```

```
Numero de hilos: 8
3.1415926536
Tiempo: 487099 ms
```

```
Numero de hilos: 16
3.1415926536
Tiempo: 487278 ms
```

## Conclusiones:

En esta practica aprendí a usar hilos, esto aplicando lo que ya conocia sobre apuntadores, aprendi como hacer que ejecuten una función y como se tiene que usar los tipos de datos con el fin de que se tenga una sintaxis valida. Ademas aprendí a distribuir en varios hilos el trabajo que se realizaria en uno en un programa normal.

Un problema que encontre fue que se tenia que revisar de forma correcta los tipos de datos que se utilizaban, ya que al trabajar con numeros tan grandes el usar, por ejemplo, un unsigned int no te permitia trabajar con los rangos de valores que se iban a necesitar, por lo que se tuvo que recurrir al long o al long long para poder asegurar que no ocurririan cosas extrañas.

Algo que no se pudo solucionar dentro de lo que esperaba del programa fue que el aumento del numero de hilos realmente no mejoro el desempeño. El programa secuencial comparado con el que tiene hilos si es mas lento, por lo que se ve que el uso de hilos realmente funciona, sin embargo, por lo que entendí, al incrementar el número de hilos tambien debería mejorar el desempeño y bajar el tiempo de ejecución, pero esto no ocurrió así, incluso en casos el tener más hilos era un poco más lento, sin embargo se podria decir que fue practicamente el mismo tiempo.