

# 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: Franco García Fernando

Tlaquepaque, Jalisco, 4 de junio de 2018.

Todas las figuras e imágenes deben tener un título y utilizar una leyenda que incluya número de la imagen ó figura y una descripción de la misma. Adicionalmente, debe de existir una referencia a la imagen en el texto.

La documentación de pruebas implica:

- 1) Descripción del escenario de cada prueba
- 2) Ejecución de la prueba
- 3) Descripción y análisis de resultados.

## Instrucciones para entrega de tarea

Esta tarea, como el resto, es **IMPRESINDIBLE** 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 10 pts
- 60% para la funcionalidad 60 pts
- 30% para las pruebas 20 pts

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.

La documentación de pruebas implica:

- 1) Descripción del escenario de cada prueba
- 2) Ejecución de la prueba
- 3) Descripción y análisis de resultados.

## 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.

$$\pi = 4 \left( \sum_{n=1}^{\infty} \left( \frac{(-1)^{(n+1)}}{(2n-1)} \right) \right)$$
$$= \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \right)$$

## 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 serie.
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	1115801
2	577473
4	345679
8	305242
16	299707

### 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)

```
/*
 * tarea_2.c
 *
 * Created on: 6 jun. 2018
 * Author: Fernando Franco
 */
#include<stdio.h>
#include<stdlib.h>
#include<windows.h>
#include<time.h>

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

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

    return 0;
}
```

### Código fuente de la versión paralelizada

```
/*
 * hilos_tarea.c
 *
 * Created on: 7 jun. 2018
 * Author: Fernando Franco
 */
```

```

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

typedef struct {
    long long int inicio;
    long long int fin;
}pi;

DWORD WINAPI PI(void *p);
double suma=0;
int main(){
    setvbuf(stderr, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);

    long long h=50000000000;
    HANDLE count[100];
    pi P[100];
    int x;
    printf("¿Cuantos hilos quieres?[2],[4],[8],[16]\n");
    scanf("%d", &x);
    h=h/x;
    clock_t start=clock();
    for( int i=0;i<x;i++){
        P[i].inicio=(h*i)+1;
        P[i].fin=h*(i+1);
        count[i]=CreateThread(NULL, 0, PI, (void
*)&P[i],0,NULL);
    }
    for(int j=0; j<x;j++){
        WaitForSingleObject(count[j], INFINITE);
    }
    clock_t stop= clock();
    long tiempo=abs(1000*(stop-start)/CLOCKS_PER_SEC);
    printf("Tiempo: %li milisegundos\n",tiempo);
    printf("PI: %.10f\n", suma);
    return 0;
}

DWORD WINAPI PI(void *p){
    pi *I=(pi*)p;
    long long i;
    double acum=0;
    for( i=I->inicio;i<=I->fin;i++)
        {

```

```

        acum+=((i+1) & 1 ? -1.0 : 1.0)/(2*i-1);
    }
    acum=acum*4;
    suma+=acum;
    return 0;
}

```

## Ejecución

Programación con Memoria Dinámica - tarea\_2.1/hilos\_tarea.c - Eclipse

```

File Edit Source Refactor Navigate Search Project Run Window Help

```

Project Explorer: Alumno, arreglos\_dinamicos, ejercicio\_Memoria\_dinamica, Estructuras\_com\_MD, examen, Hilos, Memoria dinamica, tarea\_1, tarea\_2, tarea\_2.1, TDA\_MATRIX, tipos\_datos\_abstractos

Editor: \*hilos\_tarea.c\*

```

23 clock_t start=clock();
24 for(int i=0; i<x; i++){
25     P[i].inicio=(h*i)+1;
26     P[i].fin=h*(i+1);
27     count[i]=CreateThread(NULL, 0, PI, (void *)&P[i], 0, NULL);
28 }
29 for(int j=0; j<x; j++){
30     WaitForSingleObject(count[j], INFINITE);
31 }
32 clock_t stop= clock();
33 long tiempo=abs(1000*(stop-start)/CLOCKS_PER_SEC);
34 printf("Tiempo: %li milisegundos\n", tiempo);
35 printf("PI: %.10f\n", suma);
36 return 0;
37 }

```

Problems: 0. Console: <terminated> (exit value 0) tarea\_2.1.exe [C:/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea\_2.1\Debug\tarea\_2.1.exe (7/6/18 14:43)

```

¿Cuantos hilos quieres?[2],[4],[8],[16]
1
Tiempo: 1115801 milisegundos
PI: 3.1415926536

```

Programación con Memoria Dinámica - tarea\_2.1/hilos\_tarea.c - Eclipse

```

File Edit Source Refactor Navigate Search Project Run Window Help

```

Project Explorer: Alumno, arreglos\_dinamicos, ejercicio\_Memoria\_dinamica, Estructuras\_com\_MD, examen, Hilos, Memoria dinamica, tarea\_1, tarea\_2, tarea\_2.1, TDA\_MATRIX, tipos\_datos\_abstractos

Editor: \*hilos\_tarea.c\*

```

24 long long h=50000000000;
25 HANDLE count[16];
26 pi P[16];
27 int x;
28 printf("¿Cuantos hilos quieres?[2],[4],[8],[16]\n");
29 scanf("%d", &x);
30 h=h/x;
31 clock_t start=clock();
32 for(int i=0; i<x; i++){
33     P[i].inicio=(h*i)+1;
34     P[i].fin=h*(i+1);
35     count[i]=CreateThread(NULL, 0, PI, (void *)&P[i], 0, NULL);
36 }
37 for(int j=0; j<x; j++){
38     WaitForSingleObject(count[j], INFINITE);
39 }

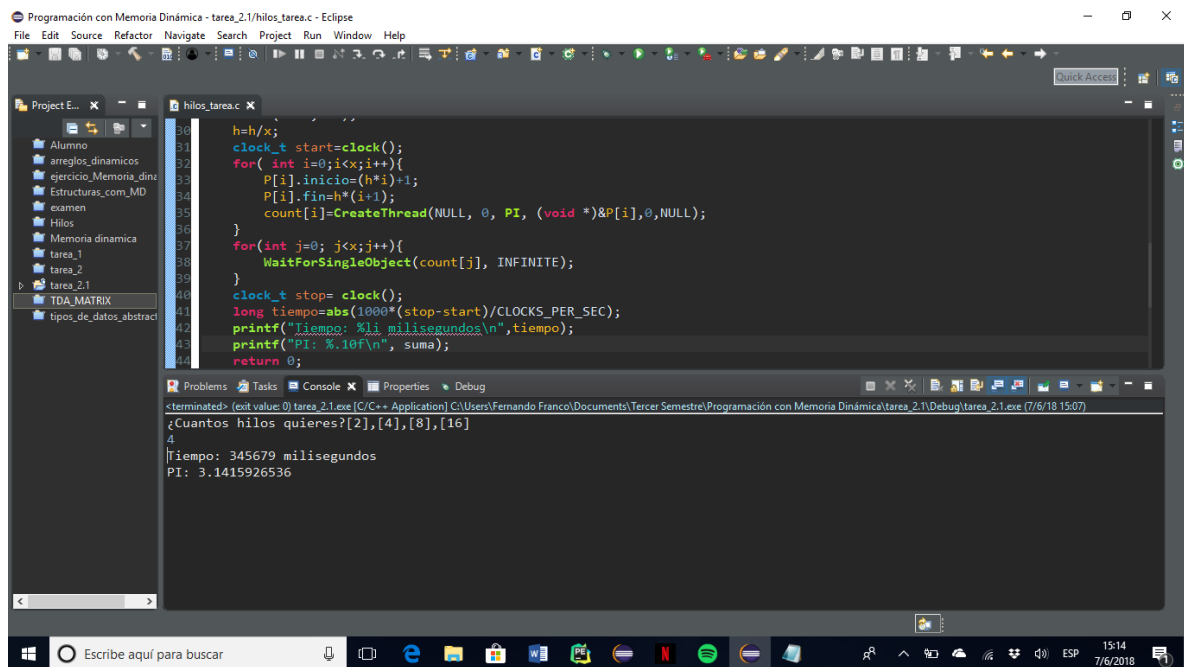
```

Problems: 0. Console: <terminated> (exit value 0) tarea\_2.1.exe [C:/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea\_2.1\Debug\tarea\_2.1.exe (7/6/18 14:24)

```

¿Cuantos hilos quieres?[2],[4],[8],[16]
2
Tiempo: 577473 milisegundos
PI: 3.1415926536

```



Programación con Memoria Dinámica - tarea\_2.1/hilos\_tarea.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer: Alumno, arreglos\_dinamicos, ejercicio\_Memoria\_dinamica, Estructuras\_com\_MD, examen, Hilos, Memoria dinamica, tarea\_1, tarea\_2, tarea\_2.1, TDA\_MATRIX, tipos\_de\_datos\_abstractos

hilos\_tarea.c

```
30 h=h/x;
31 clock_t start=clock();
32 for( int i=0;i<x;i++){
33     P[i].inicio=(h*i)+1;
34     P[i].fin=h*(i+1);
35     count[i]=CreateThread(NULL, 0, PI, (void *)&P[i],0,NULL);
36 }
37 for(int j=0; j<x;j++){
38     WaitForSingleObject(count[j], INFINITE);
39 }
40 clock_t stop= clock();
41 long tiempo=abs(1000*(stop-start)/CLOCKS_PER_SEC);
42 printf("Tiempo: %li milisegundos\n",tiempo);
43 printf("PI: %.10f\n", suma);
44 return 0;
```

Problems Tasks Console Properties Debug

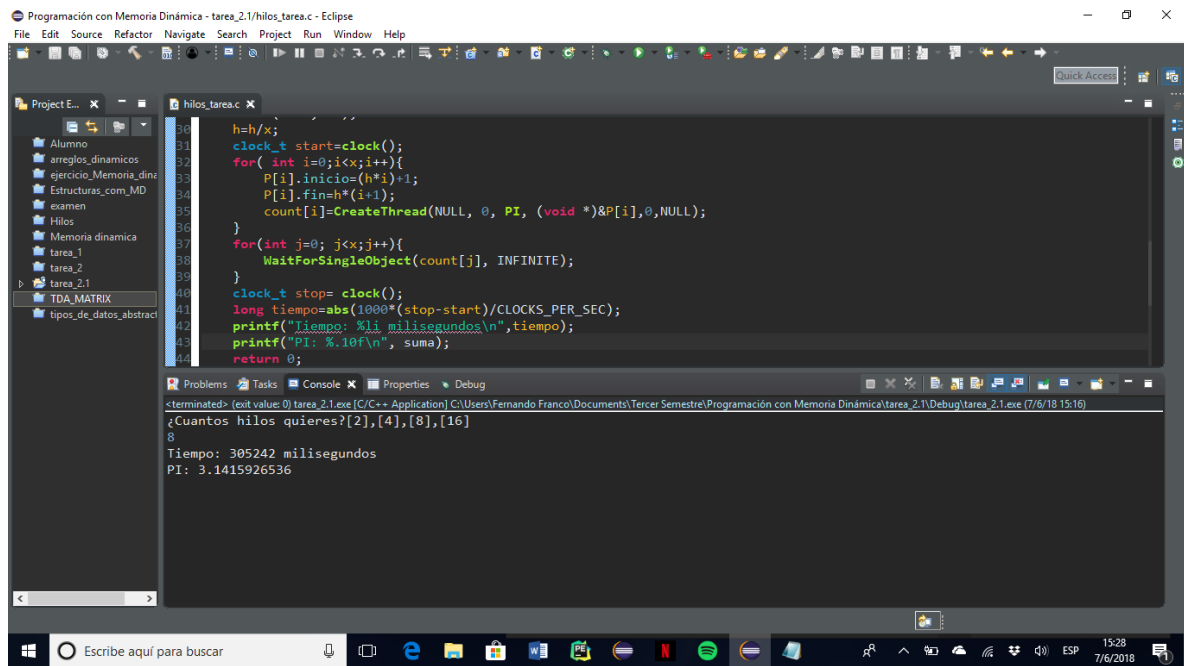
<terminated> (exit value: 0) tarea\_2.1.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea\_2.1\Debug\tarea\_2.1.exe (7/6/18 15:07)

¿Cuántos hilos quieres?[2],[4],[8],[16]

4

Tiempo: 345679 milisegundos

PI: 3.1415926536



Programación con Memoria Dinámica - tarea\_2.1/hilos\_tarea.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer: Alumno, arreglos\_dinamicos, ejercicio\_Memoria\_dinamica, Estructuras\_com\_MD, examen, Hilos, Memoria dinamica, tarea\_1, tarea\_2, tarea\_2.1, TDA\_MATRIX, tipos\_de\_datos\_abstractos

hilos\_tarea.c

```
30 h=h/x;
31 clock_t start=clock();
32 for( int i=0;i<x;i++){
33     P[i].inicio=(h*i)+1;
34     P[i].fin=h*(i+1);
35     count[i]=CreateThread(NULL, 0, PI, (void *)&P[i],0,NULL);
36 }
37 for(int j=0; j<x;j++){
38     WaitForSingleObject(count[j], INFINITE);
39 }
40 clock_t stop= clock();
41 long tiempo=abs(1000*(stop-start)/CLOCKS_PER_SEC);
42 printf("Tiempo: %li milisegundos\n",tiempo);
43 printf("PI: %.10f\n", suma);
44 return 0;
```

Problems Tasks Console Properties Debug

<terminated> (exit value: 0) tarea\_2.1.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea\_2.1\Debug\tarea\_2.1.exe (7/6/18 15:16)

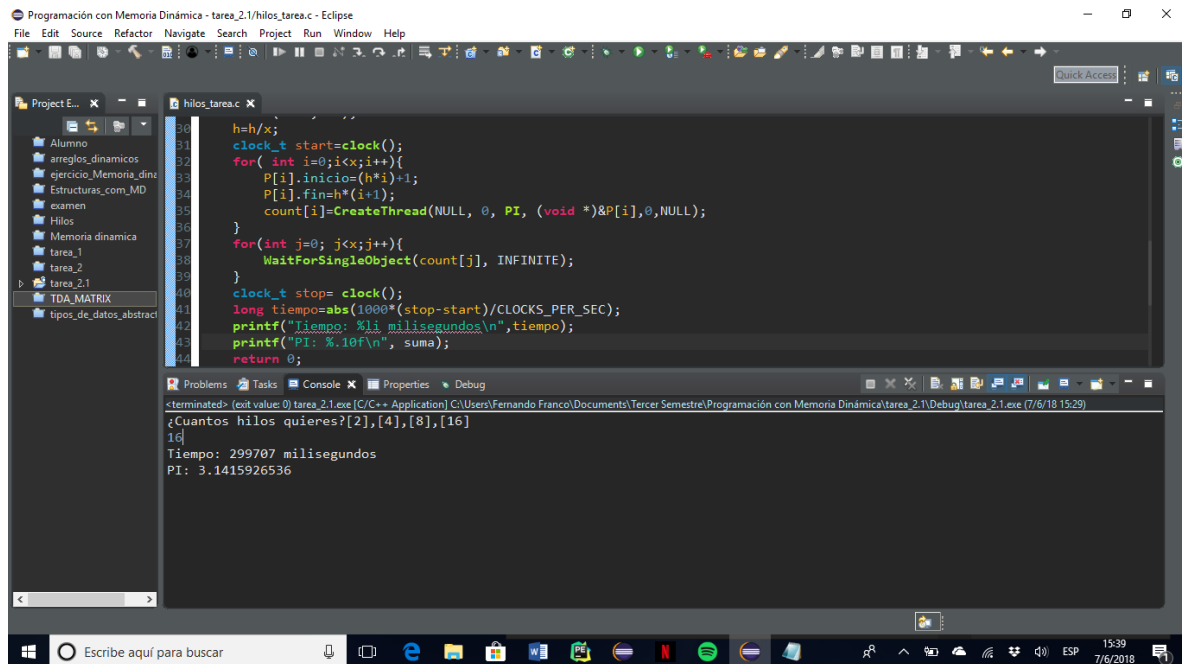
¿Cuántos hilos quieres?[2],[4],[8],[16]

8

Tiempo: 305242 milisegundos

PI: 3.1415926536





```
h=h/x;
clock_t start=clock();
for( int i=0;i<x;i++){
    P[i].inicio=(h*i)+1;
    P[i].fin=h*(i+1);
    count[i]=CreateThread(NULL, 0, PI, (void *)&P[i],0,NULL);
}
for(int j=0; j<x;j++){
    WaitForSingleObject(count[j], INFINITE);
}
clock_t stop= clock();
long tiempo=abs(1000*(stop-start)/CLOCKS_PER_SEC);
printf("Tiempo: %li milisegundos\n",tiempo);
printf("PI: %.10f\n", suma);
return 0;
```

<terminated> (exit value: 0) tarea\_2.1.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea\_2.1\Debug\tarea\_2.1.exe (7/6/18 15:29)

¿Cuántos hilos quieres?[2],[4],[8],[16]

16

Tiempo: 299707 milisegundos

PI: 3.1415926536

## Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
- ✓ Lo que me costó trabajo y cómo lo solucioné.
- ✓ Lo que no pude solucionar.

Aprendí como utilizar los hilos de una manera correcta y adecuada, aprendí para que sirven los hilos y en como estos hacen que un programa sea mucho más rápido, útil y eficiente, logre poner en practica todo lo que aprendí durante la clase de hilos.

Aun que me costó trabajo implementar los hilos, puesto que no lo vimos muy a fondo y no estaba tan seguro como usarlos ni cómo funcionaban, recibí ayuda de parte de mis compañeros que me explicaron una manera de utilizarlos de manera correcta, y funciona, ahora ya tengo mucho más claro el manejo de hilos