

# 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: Dong Llauger Jorge Alejandro

Presentación: 10 pts.  
Funcionalidad: 60 pts.  
Pruebas: 20 pts.

11 de junio de 2018. Tlaquepaque, Jalisco,

- Hay que alinear el texto
- 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 contrastar 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 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	1297373
2	1058561
4	922794
8	594063
16	318592

### 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)

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

int main()
{
    int i;
    double nsum=0.0;
    double psum=0.0;
    double pi;
    long long int n=50000000000 ;

    int hilos;
    scanf("%d",&hilos);

    clock_t start = clock();

    for (i=3;i<=n;i+=4){
        //printf("%d ",i);
        nsum+=1/(float)i;
        //printf("%f\n",nsum);
    }

    for (i=5;i<=n;i+=4){
        // printf("%d ",i);
        psum+=1/(float)i;
        //printf("%f\n",psum);
    }

    //printf("%f\n",nsum);
    pi=4*(((nsum*-1)+psum)+ 1);

    printf("%.10f\n",pi);

    clock_t stop = clock();
    int ms = 1000 * (stop - start)/CLOCKS_PER_SEC;
    printf("%d ms", ms);

    return 0;
```

```
}
```

## Código fuente de la versión paralelizada

```
#include <stdio.h>
#include <time.h>
#include <windows.h> // solo mandar en el inicio
#include <stdlib.h>

double psum=0.0;
double nsum=0.0;
static long long int k;

DWORD WINAPI calpi(void *pn);
DWORD WINAPI calpi1(void *pn);
DWORD WINAPI calpi2(void *pn);
DWORD WINAPI calpineg(int *division);
DWORD WINAPI calpipos(int *division);

int main()
{

    long long int n=5000000000000;
    int *pn;
    pn=&n;
    int division;

    int hilos;
    printf("Hilos?\n");
    scanf("%d",&hilos);

    HANDLE h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,h13,h14,h15,h16,h17,h18,h19;

    if (hilos==1){
        h1=CreateThread(NULL, 0, calpi, (void *)n, 0, NULL);
        WaitForSingleObject(h1, INFINITE);
    }
    else if (hilos==2){
        h2=CreateThread(NULL, 0, calpi1, (void *)n, 0, NULL); // negativos
        h3=CreateThread(NULL, 0, calpi2, (void *)n, 0, NULL); // positivos

        WaitForSingleObject(h2, INFINITE);
        WaitForSingleObject(h3, INFINITE);
    }
    else if(hilos==4){
        long long int n4=250000000000;

```

```

    int *pn4;
    pn4=&n4;

    h4=CreateThread(NULL, 0, calpi1, (int *)n4, 0, NULL); // negativos
    h5=CreateThread(NULL, 0, calpi2, (int *)n4, 0, NULL);// positivos
    static k=25000000000;
    h6=CreateThread(NULL, 0, calpineg, (int *)n4, 0, NULL); // negativos
    h7=CreateThread(NULL, 0, calpipos, (int *)n4, 0, NULL);// positivos

    WaitForSingleObject(h4, INFINITE);
    WaitForSingleObject(h5, INFINITE);
    WaitForSingleObject(h6, INFINITE);
    WaitForSingleObject(h7, INFINITE);

}
else if(hilos==8){
    long long int n8=12500000000;
    int *pn8;
    pn8=&n8;

    h4=CreateThread(NULL, 0, calpi1, (int *)n8, 0, NULL); // negativos
    h5=CreateThread(NULL, 0, calpi2, (int *)n8, 0, NULL);// positivos
    k=12500000000;
    h6=CreateThread(NULL, 0, calpineg, (int *)n8, 0, NULL); // negativos
    h7=CreateThread(NULL, 0, calpipos, (int *)n8, 0, NULL);// positivos
    k=12500000000*2;
    h8=CreateThread(NULL, 0, calpineg, (int *)n8, 0, NULL); // negativos
    h9=CreateThread(NULL, 0, calpipos, (int *)n8, 0, NULL);// positivos
    k=12500000000*4;
    h10=CreateThread(NULL, 0, calpineg, (int *)n8, 0, NULL); // negativos
    h11=CreateThread(NULL, 0, calpipos, (int *)n8, 0, NULL);// positivos

    WaitForSingleObject(h4, INFINITE);
    WaitForSingleObject(h5, INFINITE);
    WaitForSingleObject(h6, INFINITE);
    WaitForSingleObject(h7, INFINITE);
    WaitForSingleObject(h8, INFINITE);
    WaitForSingleObject(h9, INFINITE);
    WaitForSingleObject(h10, INFINITE);
    WaitForSingleObject(h11, INFINITE);

}
else if(hilos==16){
    long long int n16=312500000;
    int *pn16;
    pn16=&n16;

    h4=CreateThread(NULL, 0, calpi1, (int *)n16, 0, NULL); // negativos
    h5=CreateThread(NULL, 0, calpi2, (int *)n16, 0, NULL);// positivos
    k=312500000;
    h6=CreateThread(NULL, 0, calpineg, (int *)pn16, 0, NULL); // negativos
    h7=CreateThread(NULL, 0, calpipos, (int *)pn16, 0, NULL);// positivos
    k=312500000*2;
    h8=CreateThread(NULL, 0, calpineg, (int *)n16, 0, NULL); // negativos

```

```

h9=CreateThread(NULL, 0, calpipos, (int *)n16, 0, NULL);// positivos
k=312500000*4;
h10=CreateThread(NULL, 0, calpineg, (int *)n16, 0, NULL); // negativos
h11=CreateThread(NULL, 0, calpipos, (int *)n16, 0, NULL);// positivos
k=312500000*6;
h12=CreateThread(NULL, 0, calpineg, (int *)n16, 0, NULL); // negativos
h13=CreateThread(NULL, 0, calpipos, (int *)n16, 0, NULL);// positivos
k=312500000*8;
h14=CreateThread(NULL, 0, calpineg, (int *)n16, 0, NULL); // negativos
h15=CreateThread(NULL, 0, calpipos, (int *)n16, 0, NULL);// positivos
k=312500000*10;
h16=CreateThread(NULL, 0, calpineg, (int *)n16, 0, NULL); // negativos
h17=CreateThread(NULL, 0, calpipos, (int *)n16, 0, NULL);// positivos
k=312500000*10;
h18=CreateThread(NULL, 0, calpineg, (int *)n16, 0, NULL); // negativos
h19=CreateThread(NULL, 0, calpipos, (int *)n16, 0, NULL);// positivos

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);
WaitForSingleObject(h17, INFINITE);
WaitForSingleObject(h18, INFINITE);
WaitForSingleObject(h19, INFINITE);

}

clock_t start = clock();

double pi;
pi=4*(((nsum*-1)+psum)+ 1);
printf("PI: %.10f\n",pi);

clock_t stop = clock();
int ms = 1000 * (stop - start)/CLOCKS_PER_SEC;
printf("Tiempo: %d ms\n", ms);
printf("Fin del flujo principal\n");

```

```

    return 0;
}

DWORD WINAPI calpi(void *pn){
    int i;
    double pi;

    for (i=3;i<=pn;i+=4){
        //printf("%d ",i);
        nsum+=1/(float)i;
        //printf("%f\n",nsum);
    }

    for (i=5;i<= pn;i+=4){
        // printf("%d ",i);
        psum+=1/(float)i;
        //printf("%f\n",psum);
    }

    //printf("%f\n",nsum);
    pi=4*((nsum*-1)+psum)+ 1);

    printf("PI: %.10f\n",pi);
    printf("fin del hilo\n");

return 0;
}

DWORD WINAPI calpi1(void *pn){
    long long int i;

    for (i=3;i<=pn;i+=4){
        nsum+=1/(float)i;
    }

    printf("Fin hilo 1\n");
return 0;
}

DWORD WINAPI calpi2(void *pn){
    long long int i;

    for (i=5;i<= pn;i+=4){
        psum+=1/(float)i;
    }

printf("Fin hilo 2\n");
return 0;
}

DWORD WINAPI calpineg(int *pn){
    long long int i;

    for (i=pn+k;i<=pn+k;i+=4){
        nsum+=1/(float)i;
    }
}

```



```

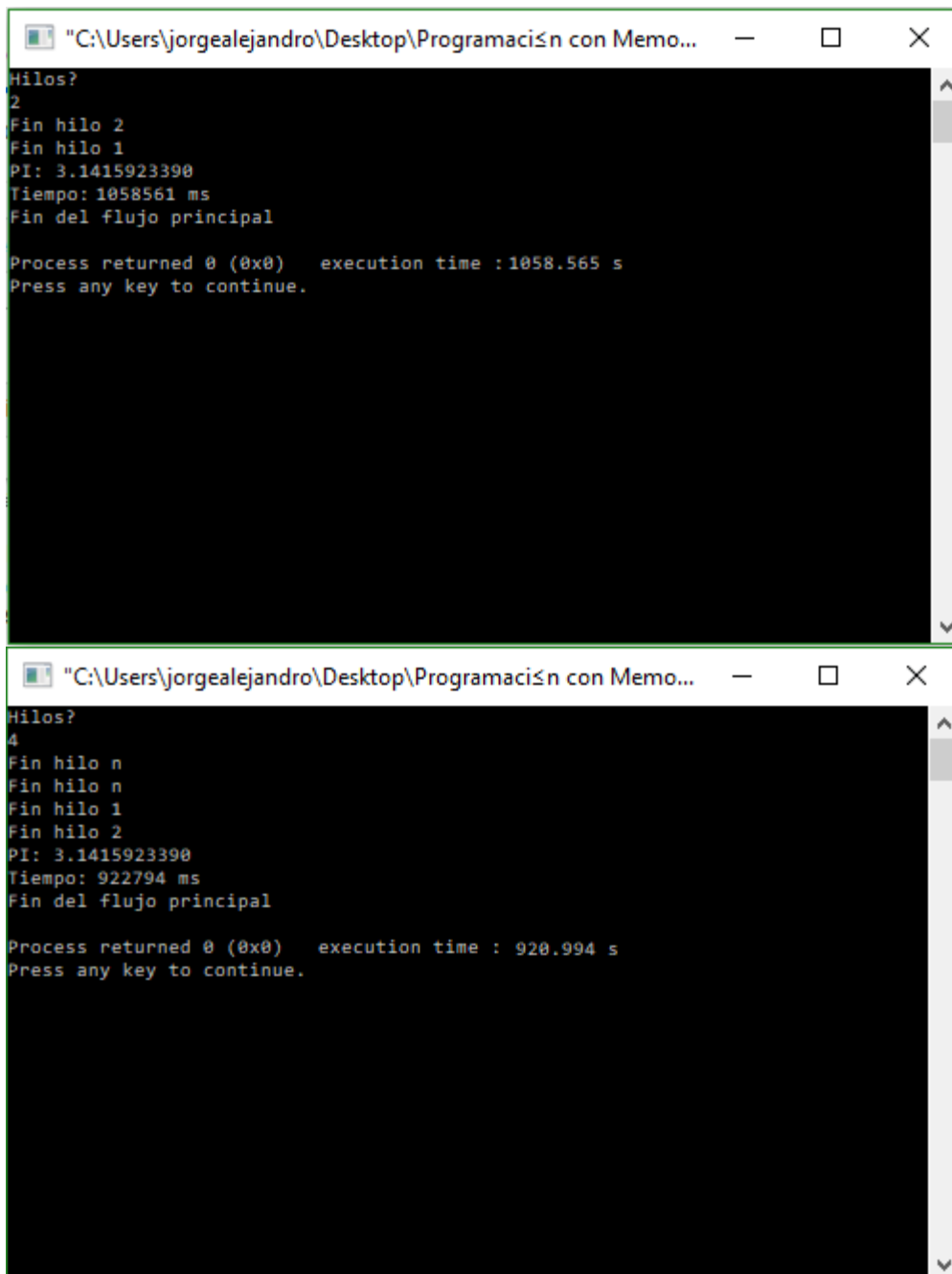
    printf("Fin hilo n\n");
return 0;
}
DWORD WINAPI calpipo(int *pn){
    long long int i;

    for (i=pn+k;i<= pn+k;i+=4){
        psum+=1/(float)i;
    }

printf("Fin hilo n\n");
return 0;
}

```

## Ejecución



```
"C:\Users\jorgealejandra\Desktop\Programación con Memo..."
Hilos?
2
Fin hilo 2
Fin hilo 1
PI: 3.1415923390
Tiempo: 1058561 ms
Fin del flujo principal

Process returned 0 (0x0)   execution time : 1058.565 s
Press any key to continue.

"C:\Users\jorgealejandra\Desktop\Programación con Memo..."
Hilos?
4
Fin hilo n
Fin hilo n
Fin hilo 1
Fin hilo 2
PI: 3.1415923390
Tiempo: 922794 ms
Fin del flujo principal

Process returned 0 (0x0)   execution time : 920.994 s
Press any key to continue.
```

```
"C:\Users\jorgealejandro\Desktop\Programaci3n con Memo...  — □ ×
Hilos?
8
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo 2
Fin hilo 1
PI: 3.1414150733
Tiempo: 594063 ms
Fin del flujo principal

Process returned 0 (0x0)   execution time : 595.785 s
Press any key to continue.
```

```
"C:\Users\jorgealejandro\Desktop\Programaci3n con Memo...  — □ ×
Hilos?
16
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo n
Fin hilo 1
Fin hilo 2
PI: 3.1414378960
Tiempo: 318592 ms
Fin del flujo principal

Process returned 0 (0x0)   execution time : 461.592 s
Press any key to continue.
```

### Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica, fue la implementación de varios hilos además de su comportamiento en diferentes situaciones, considerando los cambios que cada uno generaba al ser implementados para la optimización.
- ✓ Lo que me costó trabajo fue tener noción del comportamiento de cada hilo en la ejecución final y como lograr conjuntar el trabajo de todos el procedimiento principal lo solucioné, mediante variables que no modificaran su valor al momento de las ejecuciones.
- ✓ Lo que no pude solucionar, la optimización del código, poder crear los hilos de manera más eficiente.