

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: Manzano López de Ortigosa Luis Enrique

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 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 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 implementó 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	168578
2	86298
4	143788
8	257724
16	345152

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 <process.h>
#include <time.h>

typedef struct{
    long long inicio;
    long long fin;
    double suma;
}Rango;

double sumaPi (void *param);

int main(){

    Rango rango[]={1,5000000000,0};
    clock_t inicio, fin;

    inicio= clock();
    sumaPi(rango);
    fin=clock();
    int tiempo_final= 1000* (fin-inicio)/CLOCKS_PER_SEC;
    printf("Pi: %0.10lf \n", rango->suma);
    printf("Tiempo: %d ms\n", tiempo_final);

    return 0;
}

double sumaPi (void *param){
    Rango *aR=(Rango*)param;
    long long i;

    for(i=aR->inicio; i<=aR->fin; i++)
        if (i%2==0){
            aR->suma-= (float)4/(2*i-1);
        }
        else{
            aR->suma+= (float)4/(2*i-1);
        }
    return aR->suma;
}
```

Código fuente de la versión paralelizada

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include <process.h>
#define TERMINO 5000000000

typedef struct {
    signed long long inicio;
    signed long long fin;
    double suma;
    double sumaFinal;
}Rango;

DWORD WINAPI sumaPi (void *param);

int main(void) {
    short n;
    setvbuf(stdout, NULL, _IONBF, 0);
    printf("Hilos para calcular(1,2,4,8,16): ");
    scanf("%hi", &n);

    switch(n){
    case 1: ;
        Rango rango[]={1,TERMINO,0};
        HANDLE h1;
        clock_t inicio, fin;

        inicio= clock();
        h1=CreateThread(NULL, 0, sumaPi, (void *)&rango[0], 0, NULL);
        WaitForSingleObject(h1, INFINITE);
        fin=clock();
        int tiempo_final= 1000* (fin-inicio)/CLOCKS_PER_SEC;
        printf("Pi: %0.10lf \n", rango->suma);
        printf("Tiempo: %d ms\n", tiempo_final);

        break;

    case 2: ;
        Rango rango2[2];
        for (short i=1; i<=2; i++){
            long long m;
            m=TERMINO/2;
            if (i==1){
                rango2[i-1].inicio=1;
                rango2[i-1].fin=m;
                rango2[i-1].suma=0;
            }
            else{
                rango2[i-1].inicio=m+1;
                rango2[i-1].fin=m*i;
                rango2[i-1].suma=0;
            }
        }
    }
```

```

    }
}

double temp=0;
HANDLE h2,h3;
clock_t inicio2, fin2;
inicio2= clock();

h2=CreateThread(NULL, 0, sumaPi, (void *)&rango2[0], 0, NULL);
h3=CreateThread(NULL, 0, sumaPi, (void *)&rango2[1], 0, NULL);
WaitForSingleObject(h2, INFINITE);
WaitForSingleObject(h3, INFINITE);
fin2=clock();

int tiempo_final2= 1000* (fin2-inicio2)/CLOCKS_PER_SEC;

for (short i=0; i<2; i++){
    temp+=rango2[i].sumaFinal;
}

printf("Pi: %0.10lf \n", temp);
printf("Tiempo: %d ms\n", tiempo_final2);
break;

case 4: ;
Rango rango3[4];
for (short i=1; i<=4; i++){
    long long m;
    m=TERMINO/4;
    if (i==1){
        rango3[i-1].inicio=1;
        rango3[i-1].fin=m;
        rango3[i-1].suma=0;
    }
    else{
        rango3[i-1].inicio=m+1;
        rango3[i-1].fin=m*i;
        rango3[i-1].suma=0;
    }
}

double temp2=0;
HANDLE h4,h5,h6,h7;
clock_t inicio3, fin3;
inicio3= clock();

h4=CreateThread(NULL, 0, sumaPi, (void *)&rango3[0], 0, NULL);
h5=CreateThread(NULL, 0, sumaPi, (void *)&rango3[1], 0, NULL);
h6=CreateThread(NULL, 0, sumaPi, (void *)&rango3[2], 0, NULL);
h7=CreateThread(NULL, 0, sumaPi, (void *)&rango3[3], 0, NULL);
WaitForSingleObject(h4, INFINITE);
WaitForSingleObject(h5, INFINITE);
WaitForSingleObject(h6, INFINITE);
WaitForSingleObject(h7, INFINITE);

fin3=clock();

```

```

int tiempo_final3= 1000* (fin3-inicio3)/CLOCKS_PER_SEC;

for (short i=0; i<4; i++){
    temp2+=rango3[i].sumaFinal;
}

printf("Pi: %.10lf \n",temp2);
printf("Tiempo: %d ms\n", tiempo_final3);
break;

case 8: ;
Rango rango4[8];
for (short i=1; i<=8; i++){
    long long m;
    m=TERMINO/8;
    if (i==1){
        rango4[i-1].inicio=1;
        rango4[i-1].fin=m;
        rango4[i-1].suma=0;
    }
    else{
        rango4[i-1].inicio=m+1;
        rango4[i-1].fin=m*i;
        rango4[i-1].suma=0;
    }
}
double temp3=0;
HANDLE h8,h9,h10,h11,h12,h13,h14,h15;
clock_t inicio4, fin4;
inicio4= clock();

h8=CreateThread(NULL, 0, sumaPi, (void *)&rango4[0], 0, NULL);
h9=CreateThread(NULL, 0, sumaPi, (void *)&rango4[1], 0, NULL);
h10=CreateThread(NULL, 0, sumaPi, (void *)&rango4[2], 0, NULL);
h11=CreateThread(NULL, 0, sumaPi, (void *)&rango4[3], 0, NULL);
h12=CreateThread(NULL, 0, sumaPi, (void *)&rango4[4], 0, NULL);
h13=CreateThread(NULL, 0, sumaPi, (void *)&rango4[5], 0, NULL);
h14=CreateThread(NULL, 0, sumaPi, (void *)&rango4[6], 0, NULL);
h15=CreateThread(NULL, 0, sumaPi, (void *)&rango4[7], 0, NULL);
WaitForSingleObject(h8, INFINITE);
WaitForSingleObject(h9, INFINITE);
WaitForSingleObject(h10, INFINITE);
WaitForSingleObject(h11, INFINITE);
WaitForSingleObject(h12, INFINITE);
WaitForSingleObject(h13, INFINITE);
WaitForSingleObject(h14, INFINITE);
WaitForSingleObject(h15, INFINITE);

fin4=clock();
int tiempo_final4= 1000* (fin4-inicio4)/CLOCKS_PER_SEC;

for (short i=0; i<8; i++){
    temp3+=rango4[i].sumaFinal;
}

```

```

printf("Pi: %.10lf \n",temp3);
printf("Tiempo: %d ms\n", tiempo_final4);
break;

case 16: ;
    Rango rango5[16];
    for (short i=1; i<=16; i++){
        long long m;
        m=TERMINO/16;
        if (i==1){
            rango5[i-1].inicio=1;
            rango5[i-1].fin=m;
            rango5[i-1].suma=0;
        }
        else{
            rango5[i-1].inicio=m+1;
            rango5[i-1].fin=m*i;
            rango5[i-1].suma=0;
        }
    }
    double temp4=0;
    HANDLE
h16,h17,h18,h19,h20,h21,h22,h23,h24,h25,h26,h27,h28,h29,h30,h31;
    clock_t inicio5, fin5;
    inicio5= clock();

    h16=CreateThread(NULL, 0, sumaPi, (void *)&rango5[0], 0, NULL);
    h17=CreateThread(NULL, 0, sumaPi, (void *)&rango5[1], 0, NULL);
    h18=CreateThread(NULL, 0, sumaPi, (void *)&rango5[2], 0, NULL);
    h19=CreateThread(NULL, 0, sumaPi, (void *)&rango5[3], 0, NULL);
    h20=CreateThread(NULL, 0, sumaPi, (void *)&rango5[4], 0, NULL);
    h21=CreateThread(NULL, 0, sumaPi, (void *)&rango5[5], 0, NULL);
    h22=CreateThread(NULL, 0, sumaPi, (void *)&rango5[6], 0, NULL);
    h23=CreateThread(NULL, 0, sumaPi, (void *)&rango5[7], 0, NULL);
    h24=CreateThread(NULL, 0, sumaPi, (void *)&rango5[8], 0, NULL);
    h25=CreateThread(NULL, 0, sumaPi, (void *)&rango5[9], 0, NULL);
    h26=CreateThread(NULL, 0, sumaPi, (void *)&rango5[10], 0, NULL);
    h27=CreateThread(NULL, 0, sumaPi, (void *)&rango5[11], 0, NULL);
    h28=CreateThread(NULL, 0, sumaPi, (void *)&rango5[12], 0, NULL);
    h29=CreateThread(NULL, 0, sumaPi, (void *)&rango5[13], 0, NULL);
    h30=CreateThread(NULL, 0, sumaPi, (void *)&rango5[14], 0, NULL);
    h31=CreateThread(NULL, 0, sumaPi, (void *)&rango5[15], 0, NULL);
    WaitForSingleObject(h16, INFINITE);
    WaitForSingleObject(h17, INFINITE);
    WaitForSingleObject(h18, INFINITE);
    WaitForSingleObject(h19, INFINITE);
    WaitForSingleObject(h20, INFINITE);
    WaitForSingleObject(h21, INFINITE);
    WaitForSingleObject(h22, INFINITE);
    WaitForSingleObject(h23, INFINITE);
    WaitForSingleObject(h24, INFINITE);
    WaitForSingleObject(h25, INFINITE);
    WaitForSingleObject(h26, INFINITE);

```



```

        WaitForSingleObject(h27, INFINITE);
        WaitForSingleObject(h28, INFINITE);
        WaitForSingleObject(h29, INFINITE);
        WaitForSingleObject(h30, INFINITE);
        WaitForSingleObject(h31, INFINITE);

        fin5=clock();
        int tiempo_final5= 1000* (fin5-inicio5)/CLOCKS_PER_SEC;

        for (short i=0; i<16; i++){
            temp4+=rango5[i].sumaFinal;
        }

        printf("Pi: %.10lf \n",temp4);
        printf("Tiempo: %d ms\n", tiempo_final5);
        break;

    default:
        printf("Cantidad incorrecta. Intente de nuevo");
        break;
    }
    return EXIT_SUCCESS;
}

DWORD WINAPI sumaPi (void *param){
    Rango *aR=(Rango*)param;
    long long i;

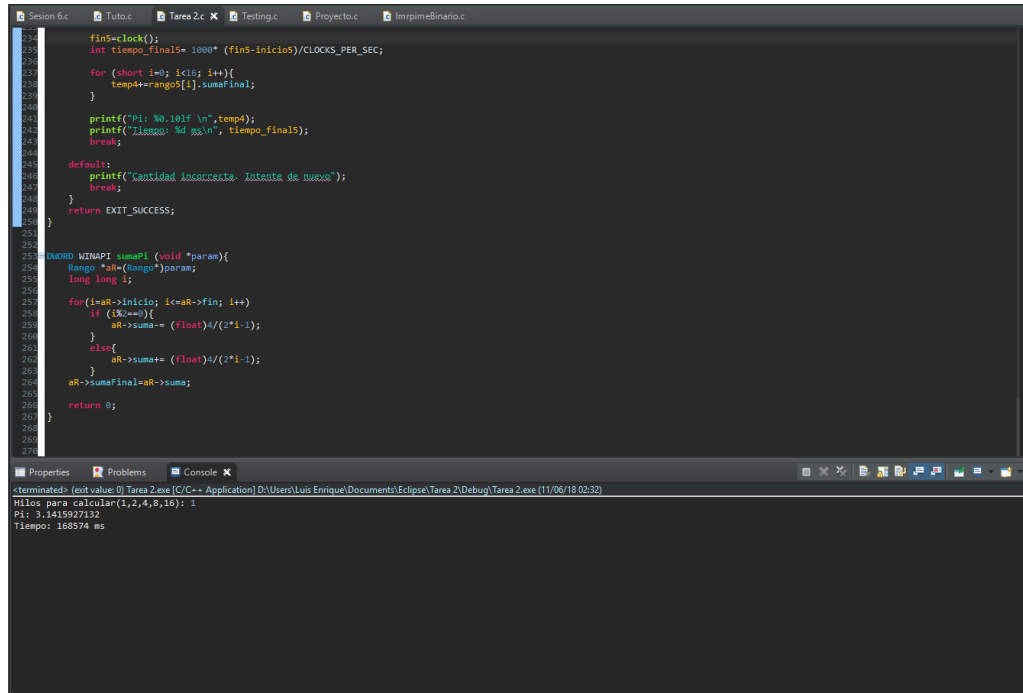
    for(i=aR->inicio; i<=aR->fin; i++)
        if (i%2==0){
            aR->suma-= (float)4/(2*i-1);
        }
        else{
            aR->suma+= (float)4/(2*i-1);
        }
    aR->sumaFinal=aR->suma;

    return 0;
}

```

Ejecución

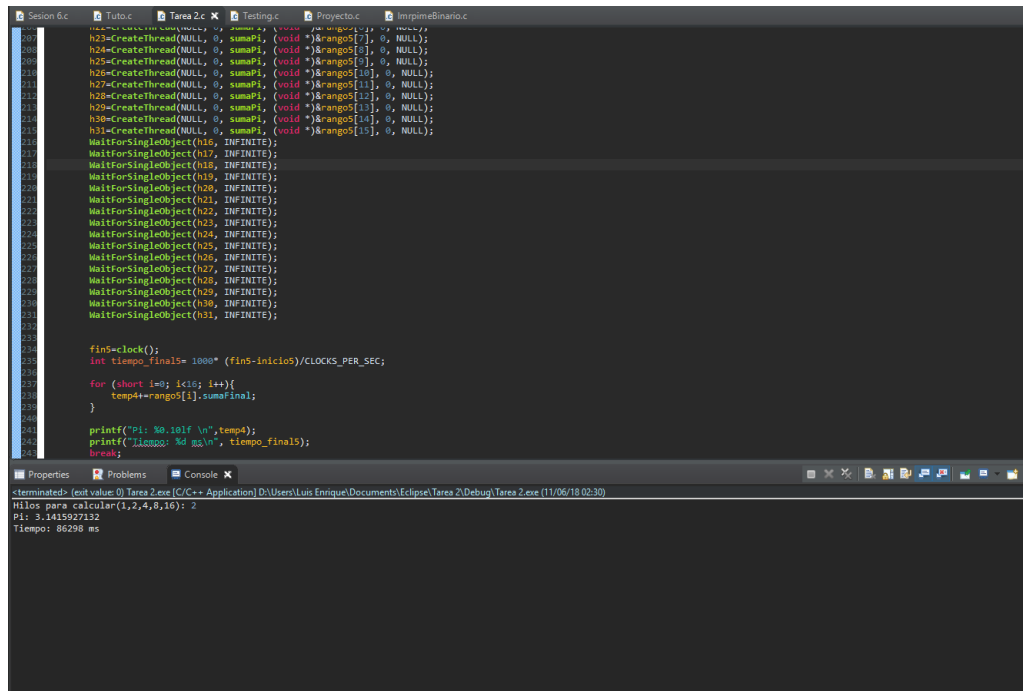
-Paralelizada: 1 hilo-



```
222     finS=clock();
223     int tiempo_finalS= 1000* (finS-inicioS)/CLOCKS_PER_SEC;
224
225     for (short i=0; i<16; i++){
226         temp4+=rangos[i].sumaFinal;
227     }
228
229     printf("Pi: %0.10lf \n",temp4);
230     printf("Tiempo: %d ms\n", tiempo_finalS);
231     break;
232
233     default:
234         printf("Cantidad incorrecta. Intenta de nuevo");
235         break;
236     }
237     return EXIT_SUCCESS;
238 }
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
Properties Problems Console
Terminated (exit value: 0) Tarea 2.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 2\Debug\Tarea 2.exe (11/06/18 02:32)
Hilos para calcular (1,2,4,8,16): 1
Pi: 3.1415927132
Tiempo: 168574 ms
```

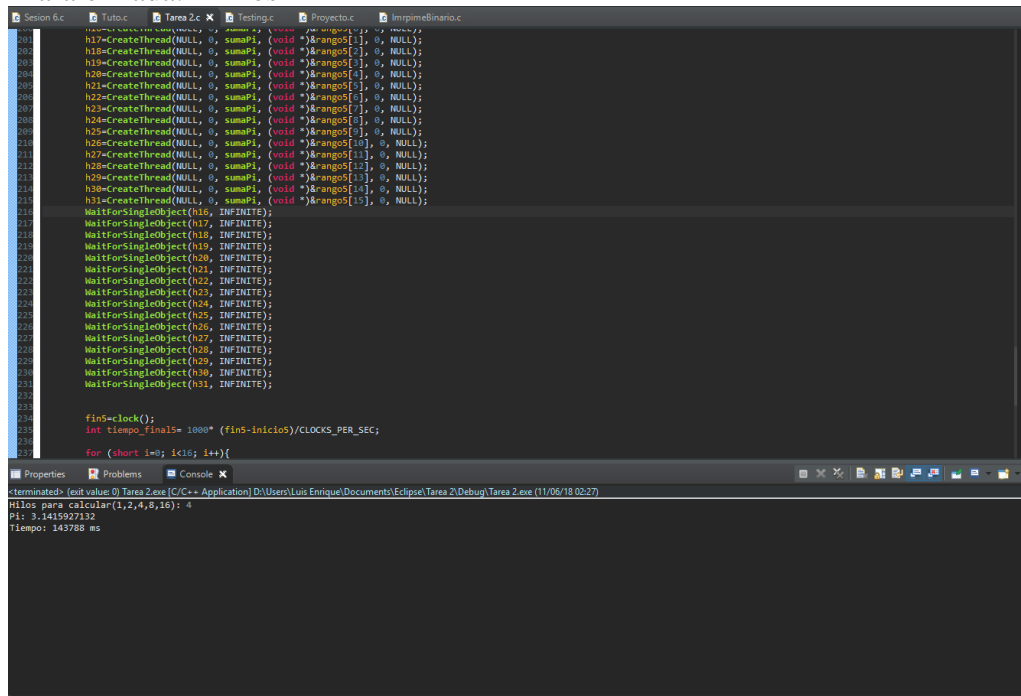
-Paralelizada: 2 hilos-



```
200     h23=CreateThread(NULL, 0, sumaPi, (void *)&rangos[2], 0, NULL);
201     h24=CreateThread(NULL, 0, sumaPi, (void *)&rangos[3], 0, NULL);
202     h25=CreateThread(NULL, 0, sumaPi, (void *)&rangos[4], 0, NULL);
203     h26=CreateThread(NULL, 0, sumaPi, (void *)&rangos[5], 0, NULL);
204     h27=CreateThread(NULL, 0, sumaPi, (void *)&rangos[6], 0, NULL);
205     h28=CreateThread(NULL, 0, sumaPi, (void *)&rangos[7], 0, NULL);
206     h29=CreateThread(NULL, 0, sumaPi, (void *)&rangos[8], 0, NULL);
207     h30=CreateThread(NULL, 0, sumaPi, (void *)&rangos[9], 0, NULL);
208     h31=CreateThread(NULL, 0, sumaPi, (void *)&rangos[10], 0, NULL);
209     h32=CreateThread(NULL, 0, sumaPi, (void *)&rangos[11], 0, NULL);
210     h33=CreateThread(NULL, 0, sumaPi, (void *)&rangos[12], 0, NULL);
211     h34=CreateThread(NULL, 0, sumaPi, (void *)&rangos[13], 0, NULL);
212     h35=CreateThread(NULL, 0, sumaPi, (void *)&rangos[14], 0, NULL);
213     h36=CreateThread(NULL, 0, sumaPi, (void *)&rangos[15], 0, NULL);
214
215     WaitForSingleObject(h16, INFINITE);
216     WaitForSingleObject(h17, INFINITE);
217     WaitForSingleObject(h18, INFINITE);
218     WaitForSingleObject(h19, INFINITE);
219     WaitForSingleObject(h20, INFINITE);
220     WaitForSingleObject(h21, INFINITE);
221     WaitForSingleObject(h22, INFINITE);
222     WaitForSingleObject(h23, INFINITE);
223     WaitForSingleObject(h24, INFINITE);
224     WaitForSingleObject(h25, INFINITE);
225     WaitForSingleObject(h26, INFINITE);
226     WaitForSingleObject(h27, INFINITE);
227     WaitForSingleObject(h28, INFINITE);
228     WaitForSingleObject(h29, INFINITE);
229     WaitForSingleObject(h30, INFINITE);
230     WaitForSingleObject(h31, INFINITE);
231
232     finS=clock();
233     int tiempo_finalS= 1000* (finS-inicioS)/CLOCKS_PER_SEC;
234
235     for (short i=0; i<16; i++){
236         temp4+=rangos[i].sumaFinal;
237     }
238
239     printf("Pi: %0.10lf \n",temp4);
240     printf("Tiempo: %d ms\n", tiempo_finalS);
241     break;
242
243     default:
244         printf("Cantidad incorrecta. Intenta de nuevo");
245         break;
246     }
247     return EXIT_SUCCESS;
248 }
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

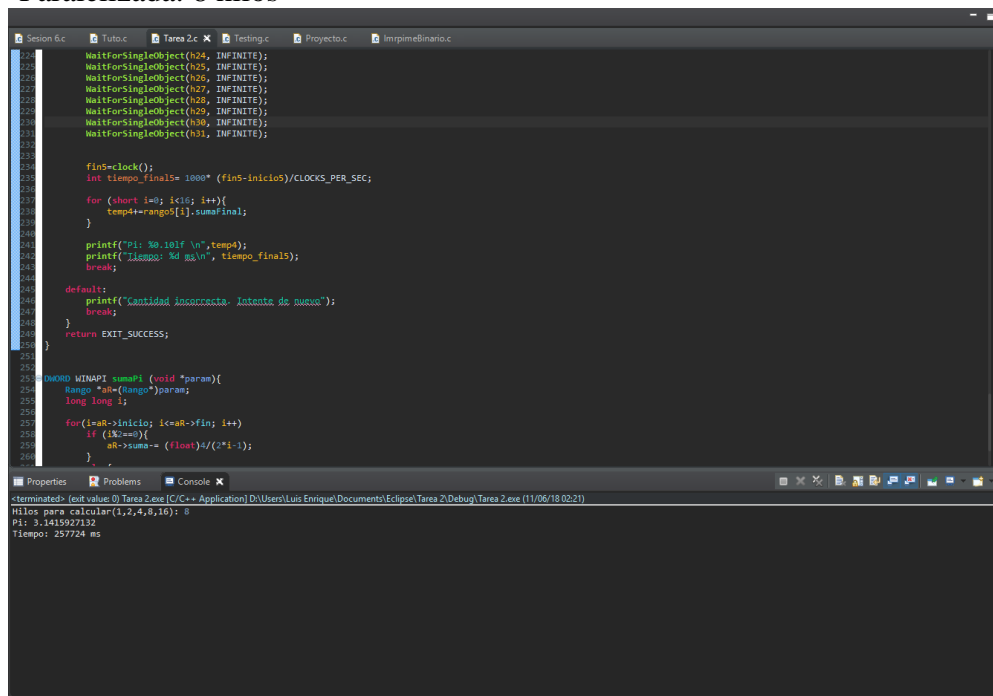
```
Properties Problems Console
Terminated (exit value: 0) Tarea 2.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 2\Debug\Tarea 2.exe (11/06/18 02:30)
Hilos para calcular (1,2,4,8,16): 2
Pi: 3.1415927132
Tiempo: 86298 ms
```

-Paralelizada: 4 hilos-



```
Session 6.c  Tutorial  Tarea 2.c  Testing.c  Proyecto.c  ImprimeBinario.c
1171 CreateThread(NULL, 0, sumaPi, (void *)&rangos[1], 0, NULL);
1172 CreateThread(NULL, 0, sumaPi, (void *)&rangos[2], 0, NULL);
1173 CreateThread(NULL, 0, sumaPi, (void *)&rangos[3], 0, NULL);
1174 CreateThread(NULL, 0, sumaPi, (void *)&rangos[4], 0, NULL);
1175 CreateThread(NULL, 0, sumaPi, (void *)&rangos[5], 0, NULL);
1176 CreateThread(NULL, 0, sumaPi, (void *)&rangos[6], 0, NULL);
1177 CreateThread(NULL, 0, sumaPi, (void *)&rangos[7], 0, NULL);
1178 CreateThread(NULL, 0, sumaPi, (void *)&rangos[8], 0, NULL);
1179 CreateThread(NULL, 0, sumaPi, (void *)&rangos[9], 0, NULL);
1180 CreateThread(NULL, 0, sumaPi, (void *)&rangos[10], 0, NULL);
1181 CreateThread(NULL, 0, sumaPi, (void *)&rangos[11], 0, NULL);
1182 CreateThread(NULL, 0, sumaPi, (void *)&rangos[12], 0, NULL);
1183 CreateThread(NULL, 0, sumaPi, (void *)&rangos[13], 0, NULL);
1184 CreateThread(NULL, 0, sumaPi, (void *)&rangos[14], 0, NULL);
1185 CreateThread(NULL, 0, sumaPi, (void *)&rangos[15], 0, NULL);
1186 WaitForSingleObject(h17, INFINITE);
1187 WaitForSingleObject(h18, INFINITE);
1188 WaitForSingleObject(h19, INFINITE);
1189 WaitForSingleObject(h20, INFINITE);
1190 WaitForSingleObject(h21, INFINITE);
1191 WaitForSingleObject(h22, INFINITE);
1192 WaitForSingleObject(h23, INFINITE);
1193 WaitForSingleObject(h24, INFINITE);
1194 WaitForSingleObject(h25, INFINITE);
1195 WaitForSingleObject(h26, INFINITE);
1196 WaitForSingleObject(h27, INFINITE);
1197 WaitForSingleObject(h28, INFINITE);
1198 WaitForSingleObject(h29, INFINITE);
1199 WaitForSingleObject(h30, INFINITE);
1200 WaitForSingleObject(h31, INFINITE);
1201
1202 finS=clock();
1203 int tiempo_final= 1000* (finS-inicioS)/CLOCKS_PER_SEC;
1204
1205 for (short i=0; i<16; i++){
1206
1207 }
1208
1209 <terminated> [exit value 0] Tarea 2.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 2\Debug\Tarea 2.exe (11/06/18 02:27)
Hilos para calcular(1,2,4,8,16): 4
Pi: 3.1415927132
Tiempo: 143788 ms
```

-Paralelizada: 8 hilos-



```
Session 6.c  Tutorial  Tarea 2.c  Testing.c  Proyecto.c  ImprimeBinario.c
1221 WaitForSingleObject(h24, INFINITE);
1222 WaitForSingleObject(h25, INFINITE);
1223 WaitForSingleObject(h26, INFINITE);
1224 WaitForSingleObject(h27, INFINITE);
1225 WaitForSingleObject(h28, INFINITE);
1226 WaitForSingleObject(h29, INFINITE);
1227 WaitForSingleObject(h30, INFINITE);
1228 WaitForSingleObject(h31, INFINITE);
1229
1230 finS=clock();
1231 int tiempo_final= 1000* (finS-inicioS)/CLOCKS_PER_SEC;
1232
1233 for (short i=0; i<16; i++){
1234     temp4+=rangos[i].sumaFinal;
1235 }
1236
1237 printf("Pi: %0.10f \n",temp4);
1238 printf("Tiempo: %d ms\n", tiempo_final);
1239 break;
1240
1241 default:
1242     printf("cantidad incorrecta. Intente de nuevo");
1243     break;
1244 }
1245 return EXIT_SUCCESS;
1246 }
1247
1248 #ifndef WINAPI sumaPi (void *param){
1249     Rang *ra=(Rang*)param;
1250     long long i;
1251     for(i=ra->inicio; i<ra->fin; i++)
1252     {
1253         if (i%2==0){
1254             ra->suma+= (float)i/(2*i-1);
1255         }
1256     }
1257 }
1258
1259 <terminated> [exit value 0] Tarea 2.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Tarea 2\Debug\Tarea 2.exe (11/06/18 02:21)
Hilos para calcular(1,2,4,8,16): 8
Pi: 3.1415927132
Tiempo: 25724 ms
```

-Paralelizada: 16 hilos-

The screenshot shows the Eclipse IDE with a C++ project named 'Tarea 2.0'. The editor displays a file named 'ImpimeBinario.c' with the following code:

```

221 WaitForSingleObject(h21, INFINITE);
222 WaitForSingleObject(h22, INFINITE);
223 WaitForSingleObject(h23, INFINITE);
224 WaitForSingleObject(h24, INFINITE);
225 WaitForSingleObject(h25, INFINITE);
226 WaitForSingleObject(h26, INFINITE);
227 WaitForSingleObject(h27, INFINITE);
228 WaitForSingleObject(h28, INFINITE);
229 WaitForSingleObject(h29, INFINITE);
230 WaitForSingleObject(h30, INFINITE);
231 WaitForSingleObject(h31, INFINITE);
232
233
234
235
236 finS=clock();
237 int tiempo_finalS= 1000* ((finS-inicioS)/CLOCKS_PER_SEC);
238
239
240 for (short i=0; i<16; i++){
241     temp4+=rangos[i].sumaFinal;
242 }
243
244 printf("P1: %0.10lf \n",temp4);
245 printf("Tiempo: %d ms\n", tiempo_finalS);
246 break;
247
248
249 default:
250     printf("Cantidad Incorrecta. Intenta de nuevo");
251     break;
252 }
253 return EXIT_SUCCESS;
254 }
255
256 DWORD WINAPI sumaPl (void *param){
257     Rango *r=Rango*(Rango*)param;
258     long long i;
259
260     for (i=r->inicio; i<=r->fin; i++)
    
```

The console output at the bottom shows the program's execution results:

```

Properties Problems Console x
x Terminated (exit value 0) Tarea 2.0.exe (C:/++ Application) D:\Users\Luis Enrique\Documents\Eclipse\Tarea 2\Debug\Tarea 2.0.exe (11/06/18 02:14)
Hilos para calcular(1,2,4,8,16): 16
P1: 3.1415927132
Tiempo: 345152 ms
    
```

-Secuencial-

```

1 //Tarea 2.c
2 //Enrique Luis
3 #include <stdio.h>
4 #include <time.h>
5
6 }Rango;
7
8 double sumaPi (void *param);
9
10
11 int main(){
12
13     Rango rango[]={1,5000000,0};
14     clock_t inicio, fin;
15
16     inicio= clock();
17     sumaPi(rango);
18     fin=clock();
19     int tiempo_final= 1000* (fin-inicio)/CLOCKS_PER_SEC;
20     printf("Pi: %e\n", rango->suma);
21     printf("Tiempo: %d ms\n", tiempo_final);
22
23
24     return 0;
25 }
26
27 double sumaPi (void *param){
28     Rango *aR=(Rango*)param;
29     long long i;
30
31     for(i=aR->inicio; i<=aR->fin; i++)
32         if (i%2==0){
33             aR->suma-= (float)i/(2*i-1);
34         }
35         else{
36             aR->suma+= (float)i/(2*i-1);
37         }
38     return aR->suma;
39 }
40
41 double sumaFinal(double array[], int cant){
42     double temp;
43
44
45 }

```

Properties Problems Console X

terminated- (exit value: 0) Testing.exe [C/C++ Application] D:\Users\Luis Enrique\Documents\Eclipse\Testing\Debug\Testing.exe (11/06/18 04:05)

Pi: 3.1415924536
Tiempo: 190 ms

Conclusiones:

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
 - Pude reconocer un poco más el razonamiento y funcionamiento de cómo los procesos son creados, desarrollados y ejecutados a través de diferentes programas.
 - Cómo los hilos bien implementados ayudan a tener una mayor eficiencia en diversos métodos y cómo estos ayudan a tener un menor tiempo de espera.
- ✓ Lo que me costó trabajo y cómo lo solucioné.
 - Me costó al principio el poder plasmar y desarrollar en código la serie de Gregory-Leibniz.
 - Además, fue un poco complejo el poder encontrar la forma de guardar los valores obtenidos al llamar la función, así como el poder ir sumando sus valores para tener el resultado final.
- ✓ Lo que no pude solucionar.
 - Hasta el momento, pude satisfactoriamente resolver todo.