

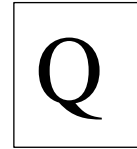
Examen 1

Nombre: Rolando Quispe Mamani

CI: 4886089 LP

Materia: INF-317 Sistemas en Tiempo Real y Distribuido

Docente: Lic. Moises Silva



Código fuente en github <https://github.com/rolandex25g/exameninf317>

1. En OPENMP cree un programa que encripte su nombre mediante CESAR

Para el cifrado CESAR se utiliza un alfabeto de letras y un desplazamiento "clave k". Con el valor de k se rota a izquierda el alfabeto para luego cifrar el mensaje letra por letra reemplazando su valor con el nuevo alfabeto rotado.

mensaje			ROLANDO																									
clave k			3																									
ALFABETO I	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
Posición	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
ALFABETO II	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C		
mensaje			R	O	L	A	N	D	O																			
cifrado			U	R	O	D	Q	G	R																			

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <omp.h>

//Busca el caracter indicado por la posicion "pos", de la cadena mensaje en el
alfabeto y
//guarda en esa misma posicion en la cadena cifrado, el caracter
correspondiente del alfabeto2
void buscar(int pos, int tamalfabeto, char *mensaje, char *cifrado, char
*alfabeto, char *alfabeto2) {
    int j;
    for (j=0; j< tamalfabeto; j=j+1) {
        if(mensaje[pos]==alfabeto[j]) {
            cifrado[pos] = alfabeto2[j];
        }
    }
}

int main ()
{
    char alfabeto[]="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char alfabeto2[]="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```

char mensaje[]="ROLANDO";
char cifrado[sizeof(mensaje)];
strcpy(cifrado,mensaje);
int k=3;//Clave de cifrado
int i,j;
int tammensaje=sizeof(mensaje)-1;
int tamalfabeto=sizeof(alfabeto)-1;

//Contruye alfabeto2 para cifrado, rotando k elementos a izquierda
for (i=0,j=k;j< tamalfabeto; j=j+1,i=i+1){
    alfabeto2[i]=alfabeto[j];
}
for (i=tamalfabeto-k,j=0;j< k;j=j+1,i=i+1){
    alfabeto2[i]=alfabeto[j];
}

printf("\n Clave cifrado %d \n",k);
printf("\n Alfabeto  %s \n",alfabeto);
printf("\n Alfabeto2 %s \n",alfabeto2);

time_t inicio,fin;
time(&inicio);

#pragma omp parallel
{
    //En cada proceso paralelo se ejecuta buscar() para una posicion i
    for (i=0;i< tammensaje; i=i+1){
        buscar(i, tamalfabeto,mensaje,cifrado,alfabeto,alfabeto2);
    }
}

printf("\n mensaje: %s \n",mensaje);
printf("\n cifrado: %s \n",cifrado);

time(&fin);
printf("\n Tiempo: %.2f segundos",difftime(fin,inicio));

return 0;
}

```

usuario1@debian10: ~

Archivo
Editar
Ver
Buscar
Terminal
Ayuda

```

root@debian10:/home/usuario1/compartir/inf317# gcc -fopenmp cesarpar.c -o cesarpar
root@debian10:/home/usuario1/compartir/inf317# ./cesarpar

Clave cifrado 3

Alfabeto  ABCDEFGHIJKLMNOPQRSTUVWXYZ

Alfabeto2  DEFGHIJKLMNOPQRSTUVWXYZABC

mensaje: ROLANDO

cifrado: URODQGR

Tiempo: 0.00 segundosroot@debian10:/home/usuario1/compartir/inf317#

```

2. En OPENMP realice la multiplicación de matrices $A=xB$ (Matrices de 100x100)

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

//Calcula el producto de matriz xB, pero solo de la fila indicada por pos
void obtenerFila(int pos,int A[][100],int x[][100],int B[][100],int ntam)
{
    int j,n;
    for (j=0;j< ntam; j=j+1){
        A[pos][j]=0;
        for (n=0;n< ntam; n=n+1){
            A[pos][j]=A[pos][j]+(x[pos][n]*(B[n][j]));
        }
    }
}

void imprimirMatriz(int mat[][100],int ntam)
{
    int i,j;
    for (i=0;i< ntam; i=i+1){
        for(j=0;j< ntam; j=j+1){
            printf("%d \t",mat[i][j]);
        }
        printf("\n");
    }
}

int main ()
{
    int i,j;

    int A[100][100];
    int x[100][100];
    int B[100][100];

    int ntam=100;
    for (i=0;i< ntam; i=i+1){
        for(j=0;j< ntam; j=j+1){
            x[i][j]=rand()%10;
            B[i][j]=rand()%10;
        }
    }

    printf("\n Matriz x \n");
    imprimirMatriz(x,ntam);

    printf("\n Matriz B \n");
    imprimirMatriz(B,ntam);

    time_t inicio,fin;
    time(&inicio);
```

```

#pragma omp parallel
{
    //En cada proceso paralelo se ejecuta obtenerFila() para una fila
i
    for (i=0; i< ntam; i=i+1) {
        obtenerFila(i,A,x,B,ntam);
    }

    printf("\n Matriz A=xB \n");
    imprimirMatriz(A,ntam);

    time(&fin);
    printf("\n Tiempo: %.2f segundos",difftime(fin,inicio));
    return 0;
}

```

```

usuario1@debian10: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@debian10:/home/usuario1/compartir/inf317# gcc -fopenmp matrizpar.c -o matrizpar
root@debian10:/home/usuario1/compartir/inf317# ./matrizpar

Matriz x
3      7      3
6      9      2
0      3      0

Matriz B
6      5      5
2      1      7
9      6      6

Matriz A=xB
59     40     82
72     51    105
6       3     21

Tiempo: 0.00 segundosroot@debian10:/home/usuario1/compartir/inf317# █

```

Ejemplo con ntam=3

3. En Python realice el cálculo de PI con multiprocessing

```

from multiprocessing import Process,Pool

def termino(n,posini,posfin,numpa):
    print("\n",n," posini:",posini," posfin:",posfin)
    paso=1/numpa
    suma=0.0
    for i in range(posini,posfin):
        x=(i+0.5)*paso

```

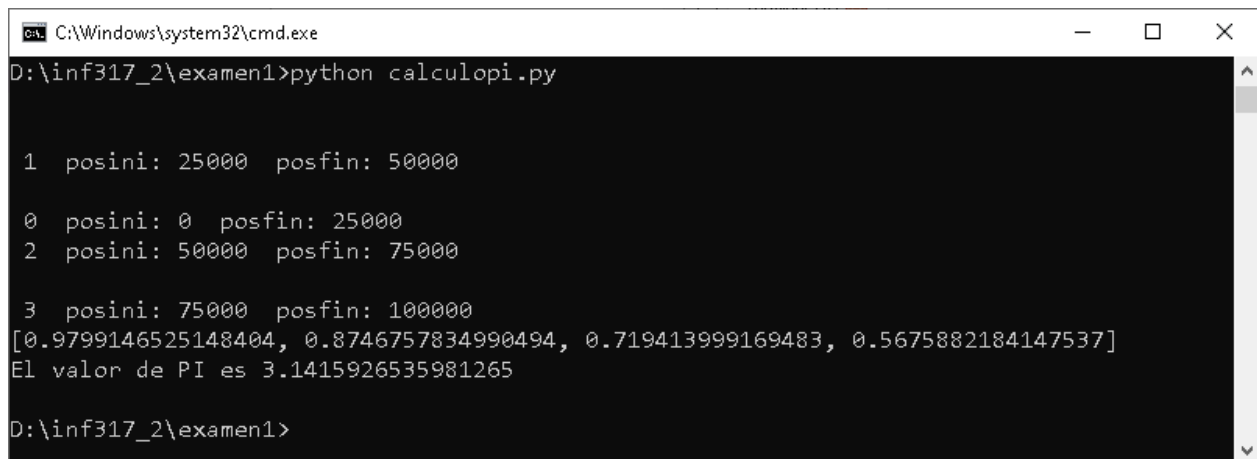
```

        suma=suma+4.0/(1.0+(x*x))
    return suma*paso

if __name__ == '__main__':
    num_pasos=100000
    with Pool(4) as p:
        lista=p.starmap(termino,[(0,0, 25000,num_pasos),
                                   (1,25000, 50000,num_pasos),
                                   (2,50000, 75000,num_pasos),
                                   (3,75000,100000,num_pasos)])

    print(lista[:])
    pi=sum(lista)
    print("El valor de PI es "+str(pi))

```



```

C:\Windows\system32\cmd.exe
D:\inf317_2\examen1>python calculopi.py

1 posini: 25000 posfin: 50000

0 posini: 0 posfin: 25000
2 posini: 50000 posfin: 75000

3 posini: 75000 posfin: 100000
[0.9799146525148404, 0.8746757834990494, 0.719413999169483, 0.5675882184147537]
El valor de PI es 3.1415926535981265

D:\inf317_2\examen1>

```

4. En c# realice el cálculo de PI con multiprocesamiento

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ConsoleApplicationCalculoPi
{
    public class GeneradorPi
    {
        public int n;
        public int posini;
        public int posfin;
        public int numpas;
        public double suma = 0;
        public void inicializar(int xn, int xposini, int xposfin, int xnumpas)
        {

```

```

        n = xn;
        posini = xposini;
        posfin = xposfin;
        numpa = xnumpa;
    }
    public void termino()
    {
        Console.WriteLine(n.ToString()+" posini:"+posini.ToString()+"
posfin:"+posfin.ToString());

        double paso = 1.0 / numpa;
        double xsuma = 0;
        double x;
        int i;
        for (i = posini; i < posfin; i = i + 1)
        {
            x = (i + 0.5) * paso;
            xsuma = xsuma + 4.0 / (1.0 + (x * x));
        }
        suma=xsuma * paso;
    }
}
class Program
{
    static void Main(string[] args)
    {
        var gen1 = new GeneradorPi();
        var gen2 = new GeneradorPi();
        var gen3 = new GeneradorPi();
        var gen4 = new GeneradorPi();
        gen1.inicializar(0, 0, 25000, 100000);
        gen2.inicializar(1, 25000, 50000, 100000);
        gen3.inicializar(2, 50000, 75000, 100000);
        gen4.inicializar(3, 75000, 100000, 100000);

        Thread hilo1 = new Thread(gen1.termino);
        Thread hilo2 = new Thread(gen2.termino);
        Thread hilo3 = new Thread(gen3.termino);
        Thread hilo4 = new Thread(gen4.termino);

        hilo1.Start();
        hilo2.Start();
        hilo3.Start();
        hilo4.Start();

        while (hilo1.IsAlive || hilo2.IsAlive || hilo3.IsAlive ||
hilo4.IsAlive)
        { }

        Console.WriteLine(gen1.suma.ToString() + ", " +
gen2.suma.ToString() + ", " + gen3.suma.ToString() + ", " +
gen4.suma.ToString());
        double pi = gen1.suma + gen2.suma + gen3.suma + gen4.suma;
        Console.WriteLine("El valor de PI es: "+pi.ToString());
    }
}

```

```

        Console.WriteLine("Presione una tecla");
        Console.ReadKey();
    }
}
}

```

