ESTUDIOS
GENERALES
CIENCIAS

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

FUNDAMENTOS DE PROGRAMACIÓN
LABORATORIO 3
PROPUESTAS DE SOLUCIÓN
SEMESTRE ACADÉMICO 2021-2

Horarios: Todos los horarios

Elaborado por Jennifer Zárate

INDICACIONES:

- Debe utilizar variables descriptivas, comentarios y mensajes descriptivos.
- El orden y la eficiencia de su implementación serán considerados en la calificación.

RESULTADOS ESPERADOS:

- Al finalizar la sesión, el alumno comprenderá el funcionamiento de la estructura algorítmica iterativa con entrada controlada.
- Al finalizar la sesión, el alumno comprenderá el funcionamiento de la iteración controlada por contador.
- Al finalizar la sesión, el alumno construirá programas usando la estructura algorítmica iterativa con entrada controlada.

CONSIDERACIONES:

- La solución presentada para cada problema corresponde a una propuesta de solución por parte del autor.
- En programación pueden existir muchas soluciones para un mismo problema pero debe cumplir con todo lo solicitado, incluyendo las restricciones brindadas.

Desarrolle los siguientes problemas en lenguaje C:

1. Suma geométrica

En Grecia, durante el siglo XIV, se usaron varias ideas para obtener el área encerrada bajo el arco de una parábola. Para ello, luego de varios estudios, se determinó que el uso de la suma infinita de términos para una sucesión permitía realizar el cálculo aproximado de este valor. En India, a través de Madhava, también existían otras ideas que tenían relación con la utilización de sucesiones y series para determinar el cálculo de diferentes funciones.

Madhava fue uno de los primeros en identificar la convergencia de una serie, que es cuando la suma de infinitos términos de una serie tiende a ser un número real. Para ello, creó métodos y realizó algunas pruebas. Por otro lado, cinco siglos después, en Europa, estos problemas se estudiaron con mucho detenimiendo por Euler y Gauss. Gauss a los 9 años identificó una forma corta de realizar el cálculo de la suma de los 100 primeros términos de una sucesión aritmética.

Luego de años, se identificó que, en matemáticas, una serie geométrica es la suma de un número infinito de términos que tiene una razón constante entre sus términos sucesivos y actualmente ya no se tiene que hacer el cálculo manual, sino que existe una fórmula obtener el valor. A continuación, se tiene la siguiente serie y su correspondiente fórmula para el cálculo de la suma geométrica:

$$1 + r + r^2 + r^3 + r^4 + ...r^{n-1} = \frac{r^n - 1}{r - 1}$$

donde:

- r es la base.
- n es la cantidad de términos.

Se le pide implementar un programa en lenguaje C que dada un cantidad de términos de la serie y una base r, calcule la suma de términos y compruebe si este valor coincide con el valor calculado por la fórmula. Deberá imprimir el valor de la suma de la serie, la suma calculada con la fórmula y un mensaje si coinciden los valores.

Nota:

Para comparar dos números decimales considere que son iguales si la diferencia entre ellos es menor o igual a un margen de error establecido de 0,0001

Caso de prueba

Caso de Prueba 1
Ingrese la cantidad de términos a evaluar: 4
Ingrese el valor de r: 4
Suma en serie: 85.000000
Suma aplicando la fórmula: 85.000000
Coincide el valor de la serie con el valor de la fórmula

Caso de Prueba 2
Ingrese la cantidad de términos a evaluar: 10
Ingrese el valor de r: 3
Suma en serie: 29524.000000
Suma aplicando la fórmula: 29524.000000
Coincide el valor de la serie con el valor de la fórmula

Caso de Prueba 3
Ingrese la cantidad de términos a evaluar: 5
Ingrese el valor de r: 6
Suma en serie: 1555.000000
Suma aplicando la fórmula: 1555.000000
Coincide el valor de la serie con el valor de la fórmula

Caso de Prueba 4
Ingrese la cantidad de términos a evaluar: 3
Ingrese el valor de r: 7
Suma en serie: 57.000000
Suma aplicando la fórmula: 57.000000
Coincide el valor de la serie con el valor de la fórmula

Programa 1: Propuesta de solución - Suma geométrica

```c
#include <stdio.h>
#include <math.h>
```

```
3
4  # define ERROR 0.0001
5
6  int main(){
7          double r, suma, sumaFormula,errorCalculado;
8          int i, nTerminos;
9
10         printf("Ingrese la cantidad de términos a evaluar: ");
11         scanf(" %d",&nTerminos);
12         printf("Ingrese el valor de r: ");
13         scanf(" %lf",&r);
14         i=1;
15         suma=0;
16         while (i<=nTerminos){
17                 suma=suma+pow(r,i-1);
18                 i++;
19         }
20         sumaFormula=(pow(r,nTerminos)-1)/(r-1);
21         errorCalculado= abs(suma-sumaFormula);
22         printf("Suma en serie: %lf\n",suma);
23         printf("Suma aplicando la fórmula: %lf\n",sumaFormula);
24         if (errorCalculado<=ERROR){
25                 printf("Coincide el valor de la serie con el valor de la fórmula\n");
26         }
27         else{
28                 printf("No coincide el valor de la serie con el valor de la fórmula\n");
29         }
30         return 0;
31 }
```

2. Mayor y menor distancia

Debido a las necesidades que el ser humano fue identificando a lo largo del tiempo, fue creando números y operaciones entre ellos así como figuras y formas. Es así que nace la geometría que estudia las figuras y fórmulas. En específico se tiene a la Geometría euclidiana y en ella se han establecido diversos cálculos para obtener valores. Por ejemplo, se tiene la distancia de un punto a una recta que es la longitud del segmento que es perpendicular a la recta y es trazado desde el punto como se indica en la siguiente figura 1:

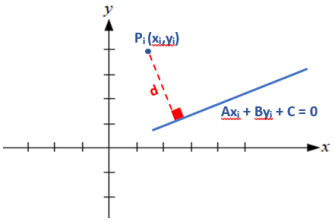


Figura 1: Distancia del punto a una recta

Para realizar el cálculo de la distancia de un punto a una recta, se utiliza la siguiente fórmula:

Recordar que:

La fórmula para calcular la distancia de un punto a una recta es:

$$distancia = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}}$$

donde:

- $x1$ y $y1$,son las coordenadas del punto 1.
- $x2$ y $y2$,son las coordenadas del punto 2.

Se le pide implementar un programa en lenguaje C que dados los coeficientes de una recta y un conjunto de puntos que serán ingresados uno a uno, pueda identificar los puntos con menor y mayor distancia a la recta y el promedio de todas las distancias calculadas.

Caso de prueba

Caso de Prueba 1
Ingrese los coeficientes A, B y C de la recta de forma Ax+By+C=0 :4 6 7
Ingrese el número de puntos a evaluar:5
Ingrese el punto 1 (x,y): 3 4
Distancia punto 1 a la recta: 5.963027
Ingrese el punto 2 (x,y): -2 1
Distancia punto 2 a la recta: 0.693375
Ingrese el punto 3 (x,y): 8 2
Distancia punto 3 a la recta: 7.072428
Ingrese el punto 4 (x,y): -5 -8
Distancia punto 4 a la recta: 8.459178
Ingrese el punto 5 (x,y): 1 1
Distancia punto 5 a la recta: 2.357476
El punto con menor distancia a la recta es (-2.00,1.00)
El punto con mayor distancia a la recta es (-5.00,-8.00)
El promedio de distancias de los puntos evaluados es: 4.91
———————————————————————————-

Caso de Prueba 2
Ingrese los coeficientes A, B y C de la recta de forma Ax+By+C=0 :5 -3 6
Ingrese el número de puntos a evaluar:4
Ingrese el punto 1 (x,y): 4 2
Distancia punto 1 a la recta: 3.429972
Ingrese el punto 2 (x,y): -3 -1
Distancia punto 2 a la recta: 1.028992
Ingrese el punto 3 (x,y): 5 4
Distancia punto 3 a la recta: 3.258473
Ingrese el punto 4 (x,y): -6 1
Distancia punto 4 a la recta: 4.630462
El punto con menor distancia a la recta es (-3.00,-1.00)
El punto con mayor distancia a la recta es (-6.00,1.00)
El promedio de distancias de los puntos evaluados es: 3.09

Programa 2: Propuesta de solución - Mayor y menor distancia

```c
#include <stdio.h>
#include <math.h>

int main(){
	int nPuntos,i;
	double A,B,C,xi,yi,xMayor,yMayor,xMenor,yMenor,distancia,distanciaMayor,distanciaMenor,suma,promedio;

	printf("Ingrese los coeficientes A, B y C de la recta de forma Ax+By+C=0 :");
	scanf(" %lf  %lf  %lf",&A,&B,&C);
	printf("Ingrese el número de puntos a evaluar:");
	scanf(" %d",&nPuntos);

	i=1;
	distanciaMayor=0;
	distanciaMenor=1000;
	suma=0;
	while (i<=nPuntos){
		printf("Ingrese el punto # %d (x,y): ",i);
		scanf(" %lf  %lf",&xi,&yi);
		/*calcular la distancia e ir buscando la mayor y menor distancia*/
		distancia=abs(A*xi+B*yi+C)/sqrt(pow(A,2)+pow(B,2));
		printf("Distancia punto  %d a la recta:  %lf\n",i,distancia);
		suma+=distancia;
		if (distancia>distanciaMayor){
			xMayor=xi;
			yMayor=yi;
			distanciaMayor=distancia;
		}
		if (distancia<distanciaMenor){
			xMenor=xi;
			yMenor=yi;
			distanciaMenor=distancia;
		}
		i++;
	}
	promedio=suma/nPuntos;

	printf("El punto con menor distancia a la recta es ( %.2lf, %.2lf)\n",xMenor,yMenor);
	printf("El punto con mayor distancia a la recta es ( %.2lf, %.2lf)\n",xMayor,yMayor);
	printf("El promedio de distancias de los puntos evaluados es:  %.2lf\n",promedio);

	return 0;
}
```

3. Cálculos de divisores

Desde hace mucho tiempo, el hombre ha tenido la necesidad de repartir cantidades de todo tipo de cosas como por ejemplo: alimentos, terrenos, herramientas, dinero, entre otros. Esta repartición se daba entre distintas personas de manera que sea en partes iguales y así cada uno tendría el mismo número de unidades.

Poco a poco y conforme transcurría el tiempo, el hombre descubrió que este inconveniente que tenía de la repartición a veces tenía solución y a veces no. Por lo que se vio en la necesidad de buscar información para resolver estos casos y así inició la divisibilidad.

La divisibilidad de los números es conocida desde hace cientos de años atrás. Por ejemplo, los egipcios ya conocían los números pares e impares. Mientras que algunas culturas como los hindúes, conocían ciertas divisibilidades por tres, sierte y nueve. Por tro lado Euclides, en Grecia, creó y demostró los teoremos básicos de la divisibilidad de números enteros; y finalmente, en el año 1623, Pascal propuso ciertas reglas para conocer la divisibilidad de cualquier número entero.

Evaluar si un número es divisor de otro, es muy importante para realizar diversos cálculos de repartición de recursos.

Se le pide implementar un programa en lenguaje C que dados dos números enteros, calcule e imprima la suma y el promedio de divisores comunes de los valores ingresados sin contar el divisor 1 ni el mismo número. Asuma que los dos números son enteros positivos. Si los números no tienen algún divisor en común, enviar el mensaje: ”Los números ingresados no tienen divisores en común.”

Caso de prueba

Caso de Prueba 1
Ingrese los dos números a evaluar: 220 20
La suma de divisores comunes entre los dos números es: 21
El promedio de divisores comunes entre los dos números es: 5.25
———————————————————————————-

Caso de Prueba 2
Ingrese los dos números a evaluar: 45 37
Los números ingresados no tienen divisores en común.
———————————————————————————-

Caso de Prueba 3
Ingrese los dos números a evaluar: 55 99
La suma de divisores comunes entre los dos números es: 11
El promedio de divisores comunes entre los dos números es: 11.00
———————————————————————————-

Caso de Prueba 4
Ingrese los dos números a evaluar: 10 600
La suma de divisores comununes entre los dos números es: 7
El promedio de divisores comununes entre los dos números es: 3.50

Programa 3: Propuesta de solución - Cálculos de divisores

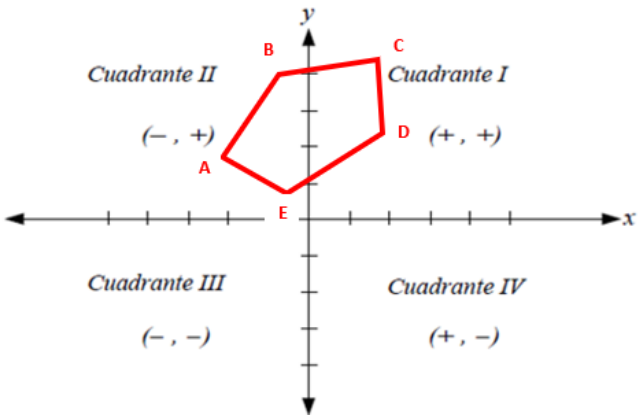
```c
#include <stdio.h>


int main(){
        int a,b,num,numMayor,i,cantDivComunes,suma;
        double promedio;

        printf("Ingrese los dos números a evaluar: ");
        scanf(" %d  %d",&a,&b);

        /*buscar el menor de ambos para trabajar con sus divisores*/
        if (a<b){
                num=a;
                numMayor=b;
        }
        else{
                num=b;
                numMayor=a;
        }
        /*buscar el menor divisor de ambos números que no incluya a 1*/
        i=2;
        cantDivComunes=0;
        suma=0;
        while (i<num){
                if ((num  % i)==0 && (numMayor  % i)== 0){
                        suma+=i;
                        cantDivComunes++;
                }
                i++;
        }
        if (cantDivComunes!=0){
                promedio=(double)suma/cantDivComunes;
                printf("La suma de divisores comunes entre los dos números es:  %d\n",suma);
```

```
34                  printf("El promedio de divisores comunes entre los dos números es:  %.2lf\n",promedio);
35          }
36          else{
37                  printf("Los números ingresados no tienen divisores en común.");
38          }
39          return 0;
40  }
```

4. Polígono Irregular

Con el paso del tiempo, el ser humano tuvo la necesidad de crear números y contar, medir y realizar cálculos. Fue así que en Egipto, empezó a imaginarse y crear conceptos que llevaban consigo formas diferentes de figuras debido a la necesidad de realizar construcciones como diques para encauzar las aguas del río Nilo y así proteger a sus cultivos. Es ahí donde se origina la geometría que significa "medida de la tierra"

Una de las formas básicas que se creó, fue el polígono, que es una figura plana geométrica que está limitada por tres o más rectas y tiene tres o más ángulos y vértices. El valor de un lado del polígono se puede calcular como la distancia entre dos vértices consecutivos del polígono.



Existen polígonos regulares cuyos lados miden el mismo valor y los polígonos irregulares cuyos lados no tienen el mismo valor.

A continuación, se le pide implementar un programa en lenguaje C que dado el número de lados del polígono, calcule el perímetro del mismo a través de las distancias de sus lados, las mismas que serán obtenidas a través del cálculo de distancia entre dos puntos. Considere que cada vértice del polígono será un punto del plano cartesiano que se representará de la forma (x,y). Todos los vértices se ingresarán de manera consecutiva. Deberá validar que los vértices se encuentren en el primer o segundo cuadrante para realizar el cálculo del perímetro. Si el usuario ingresa otro valor, deberá enviar un mensaje y solicitar que vuelva a ingresar los vértices con las especificaciones correctas. Asuma que el primer punto ingresado cumple con pertenecer al primer o segundo cuadrante.

Recordar que:

La fórmula para calcular la distancia entre dos puntos es:

$$distancia = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

donde:

- $x1$ y $y1$,son las coordenadas del punto 1.
- $x2$ y $y2$,son las coordenadas del punto 2.

Caso de prueba

Caso de Prueba 1
Ingrese el número de lados del polígono: 4
Ingrese el punto 1 (x,y): -2 1
Ingrese el punto 2 (x,y): -2 3
Ingrese el punto 3 (x,y): 2 3
Ingrese el punto 4 (x,y): 2 1
El perímetro del polígono de 4 lados es: 12.000000

Caso de Prueba 2
Ingrese el número de lados del polígono:3
Ingrese el punto 1 (x,y): -5 3
Ingrese el punto 2 (x,y): -5 -3
El punto ingresado no pertenece al primer ni al segundo cuadrante.
Ingrese el punto 2 (x,y): -5 10
Ingrese el punto 3 (x,y): 1 3
El perímetro del polígono de 3 lados es: 22.219544

Caso de Prueba 3
Ingrese el número de lados del polígono:8
Ingrese el punto 1 (x,y): 10 7
Ingrese el punto 2 (x,y): 9 5
Ingrese el punto 3 (x,y): 6 5
Ingrese el punto 4 (x,y): 4 4
Ingrese el punto 5 (x,y): 1 2
Ingrese el punto 6 (x,y): -1 -1
El punto ingresado no pertenece al primer ni al segundo cuadrante.
Ingrese el punto 6 (x,y): -1 0
Ingrese el punto 7 (x,y): 6 3
Ingrese el punto 8 (x,y): 7 6
El perímetro del polígono de 8 lados es: 27.846443

Programa 4: Propuesta de solución - Polígono irregular

```
1 #include <stdio.h>
```

```c
#include <math.h>

int main(){
	int nLados,i;
	double x1,y1,XInicial,YInicial,xi,yi,distancia,perimetro;

	printf("Ingrese el número de lados del polígono:");
	scanf(" %d",&nLados);

	/*Se solicita que ingrese el 1er punto antes de la iterativa para tener un punto con el cual realizar el cálculo de la 1ra distancia
	*/
	printf("Ingrese el punto #1 (x,y): ");
	scanf(" %lf  %lf",&x1,&y1);
	/*guardar una copia del punto inicial para realizar el cálculo de la distancia con el último punto*/
	XInicial=x1;
	YInicial=y1;

	perimetro=0;
	i=2;
	while (i<=nLados){
		printf("Ingrese el punto # %d (x,y): ",i);
		scanf(" %lf  %lf",&xi,&yi);
		/*validar punto ingresado está en el primer o segundo cuadrante*/
		if (yi>=0){
			/*calcular la distancia e ir acumulando el perimetro*/
			distancia=sqrt(pow((xi-x1),2)+pow((yi-y1),2));
			perimetro+=distancia;
			/*actualizar el valor del punto anterior*/
			x1=xi;
			y1=yi;
			i++;
		}
		else{
			printf("El punto ingresado no pertenece al primer ni al segundo cuadrante.\n ");
		}
	}
	/*falta adicionar la distancia del último al primer punto*/
	distancia=sqrt(pow((xi-XInicial),2)+pow((yi-YInicial),2));
	perimetro+=distancia;
	printf("El perímetro del polígono de  %d lados es:  %lf",nLados,perimetro);
	return 0;
}
```

Debe usar estructuras algorítmicas iterativas con entrada controlada por contador y estructuras selectivas simples o dobles.

No debe usar ningún tipo de anidamiento de estructuras algorítmicas selectivas ni programación modular.