Problema 1

Enunciado del problema

Debe ingresar un vector de 10 elementos, llenarlo de numeros pares del 2 al 20. Al iniciar el programa debe preguntar al usuario cómo quiere ver los números. El menú debe ser promedio de caracteres: "a" para verlos de forma ascendente, "d" de forma descendente, en caso que el usuario ingrese otro valor debe de decir que no es correcto y preguntar el caracter nuevamente, hasta que seal el correcto, al ingresar el valor correcto muestra el vector en pantalla y termina el programa.

Metodología para resolver el programa

El programa debe presentar un menu en donde el usuario debe elegir entre tres principales opciones:

- 'a' para ordenar en forma ascendente.
- 'd' para ordenar en forma descendente.
- 's' para salir

En caso el usuario ingrese cualquier otra letra, debe volverse a desplegar el menú. Para ello, se utiliza un switch para desplegar las acciones una vez ingresados los valores. En el caso el usuario ingrese una opción no válida, automaticamente se le dirigirá a la misma función.
Primero debe desplegarse el vector desordenado, y luego debe mostrarse la opción de ordenamiento.
Se crearán dos funciones en las cuales se ordenaá el vector según la opción elegida.
Para el ordenamiento se eligió el ordenamiento por selección.
El ordenamiento por selección busca el valor mayor a medida que hace una pasada y, después de completar la pasada, lo pone en la ubicación correcta. Después de la primera pasada, el ítem mayor está en la ubicación correcta. Después de la segunda pasada, el siguiente mayor está en su ubicación. Este proceso continúa y requiere n-1 pasadas para ordenar los n ítems, ya que el ítem final debe estar en su lugar después de la (n-1)-ésima pasada.
Cabe mencionar que para ambos casos de ordenamiento, el código es aproximadamente el mismo. Sólamente cambia el signo ¿por ¡en la siguiente pieza de código:

```
if(vector[j]<vector[min]) -> en forma ascendente
  if(vector[j]> vector[min]) -> en forma descendente
```

Entradas y salidas

Se le solicitará al usuario una variable caracter, el cual indicará si desea ordenar el vector ascendentemente o descendentemente. Al ordenarlo, se obtendrá un vector.
$\rightarrow Caracter$
$\leftarrow Vectorordenado$

Pseudocódigo

1. Crear variables globales de tipo entero, para almacenar el vector desordenado y otro para almacenar la longitud del vector
2. Mostrar el vector desordenado
3. Desplegar el menú con las opciones.
    a) a, orden ascendente
    b) d, orden descendente
    c) s, salir
4. Guardar la opción elegida en una variable tipo carácter
5. Si:
    a) a, digir a función 'ascendente'
    b) d, dirigir a función 'descendente'
    c) s, salir del ciclo
6. Sino:
    a) Mostrar: Vuelva a ingresar una opcion válida
7. En ascendente:
    a) Declarar variables enteras i,j, temp, min
    b) i=0, j=i+1
    c) Si:

    a) i< longitud
    b) min=i
    c) j< longitud
        1) si: vector[j]<vector[min]
        2) min=j

        1) j++

        1) temp=vector[i]
        2) vector[i]=vector[min]
        3) vector[min]=temp

        1) i++

    a) Mostrar vector ordenado ascendente

8. En descendente:

   a) Declarar variables enteras i,j, temp, min

   b) i=0, j=i+1

   c) Si:

   a) i¡longitud

   b) min=i

      1) j¡longitud

         a′ vector[j]> vector[min]

         b′ min=j

         c′ j++

      2) temp=vector[i]

      3) vector[i]=vector[min]

      4) vector[min]=temp

      5) i++

   c) Mostrar vector ordenado ascendente

9. Fin

código del programa

El código puede ser encontrado en el GitHub

```
Usuario: CindyGat, Repositorio LabSimu2021 laboratorio4, problema1.c
```

link:

`https://github.com/CINDYGAT/LabSimu1S2021/blob/main/Laboratorio4/problema1.c`

Problema 2

Enunciado del problema

Crear un programa que solicite al usuario 5 números enteros, estos se deben de guardar en un vector. Al terminar de guardar los valores, el programa debe ordenarlos de forma ascendente y mostrar el vector ordenado. (utilice un metodo de ordenacion.)

Metodología para resolver el programa

El problema consiste en utilizar tres funciones principales. Una de estas le permite al usuario ingresar el vector de cinco posiciones. Otro para mostrar el vector desordenado ingresado y la funcuón que ordena el vector de forma ascendente.
Con las funciones definidas, se declaran ciclos for para ingresar y mostrar vectores.
Para el ordenamiento ascendente de los vectores, se utilizó el ordenamiento por inserción. El ordenamiento consiste en seleccionar el segundo valor como clave y se lo compara con los valores ubicados a su izquierda. Si el valor es menor entonces se inserta en el lugar correspondiente. Se selecciona el siguiente número como clave y se repite el proceso para todos los valores anteriores.
Se selecciona la siguiente clave. Se sigue comparando con cada número a su izquierda hasta encontrar uno que sea menor o llegar al principio de la lista. Finalmente se selecciona la última clave.
Por ejemplo, considerar el vector: v=[5,2,7,1,6]

1. Se considera el número 2 y se compara con lo que está a su izquierda. En este caso, el 5. Ya que 5>2, entonces cambian posiciones. Se guarda temporalmente 5 en una variable temporal, y se hace el cambio de 2 a la posición de 5, luego 5 se reescribe en la posicion anterior de 2. Se obtiene → v=[2,5,7,1,6].

2. La casilla se corre 1 posición, siendo esta 7. Este debe compararse con todo lo que está a su izquierda. Primero con 5, ya que 5 y 2 <7, entonces el vector permanece igual.

3. Corriendo una casilla, vector[j]=1. Comparando con los vectores a la izquierda vector[j-1], notamos que 2,5,7>1, entonces 1 se mueve a la primera casilla. Entonces obtenemos vector=[1,2,5,7,6]

4. Corriendo una casilla, en vector[j]=6. Se compara con vector[j-1], se nota que 1,2,5 <6 y 7>6, entonces vector=[1,2,5,6,7]. EL vector está ordenado.

Se hace notar que:
j> 0 →asegurando que siempre se tomará en cuenta el vector a la derecha del primer elemento.
temp <vector[j-1] → Asegurando que la variable temporal siempre debe ir después del vector previo.

Entradas y salidas

Entrada: 5 variables de tipo entero que conformarán un vector
Salida: Vector compuesto de 5 valores enteros ordenados ascendentemente
→ 5 variables de tipo entero ← Vector de 5 posiciones ordenado

Pseudocódigo

1. Inicio
2. Crear prototipos de funciones
3. En función IngresoVector
   - *a*) Definir enteros vector 5 posiciones, longitud, i
   - *b*) longitud =5
   - *c*) i=0
   - *d*) i< longitud
     1) vector[i]=valor ingresado
     2) i++
   - *e*) Dirigirse a función MostrarVector
   - *f*) Dirigirse a función OrdenarVector
4. En función Mostrar vector
   - *a*) Entero vector[5], longitud,i
   - *b*) i=0
   - *c*) Si i< longitud
     1) Imprimir vector[i]
     2) i++
5. En función Ordenar vector
   - *a*) Entero i,j,temp, vector[5], longitud
   - *b*) i=0
     1) Entero i,j,temp, vector[5], longitud
     2) i=0
        - *a′* si i < longitud;
        - *b′* j=i
        - *c′* temp=vector[i]
        - *d′* Mientras j< 0 y temp<vector[j-1]
        - *e′* vector[j]=vector[j-1]
        - *f′* j–
        - *g′* vector[j]=temp

Código del programa

El código puede ser encontrado como:

```
Usuario: CindyGat, Repositorio LabSimu2021 laboratorio4, problema2.c
```

link:

https://github.com/CINDYGAT/LabSimu1S2021/blob/main/Laboratorio4/Problema2.c

Problema 3

Enunciado del problema

Crear un programa que solicite al usuario dos posiciones en coordenadas (x,y,z) al obtenerlas debe almacenarlas en dos vectores, el programa automaticamente debe de mostrar los siguientes resultados:
a) magnitud de cada vector
b) suma de los dos vectores
c) producto escalar
d) producto vectorial

Metodología para resolver el problema

El programa debe ser capaz de calcular la magnitud de los vectores ingresado, la suma, El producto escalar y vectorial. Para ello, se utilizan cinco funciones, las cuales se encargan de las operaciones anterior, más una función extra, que se encarga de desplegar los vectores ingresados por el usuario.

Para el ingreso y despliegue de los vectores, se utiliza la forma habitual con un ciclo for, el cual itera hasta alcanzar un tamaño igual al de su longitud, en este caso 3.

Para la magnitud de los vectores, se utiliza una sola funciones a la cual le ingresamos distintos valores como parámetros.
Para realizar las operaciones, se utiliza la fórmula:

$$(x, y, z) = \sqrt{(}A_x^2 + A_y^2 + A_z^2)$$

Este valor, al ser un número tipo float, entonces puede ser retornado.

Para la suma, se sigue la regla usual de esta. Se suman las componentes. Ya que este es un vector, se opta por imprimirlo en la misma función, y llamarlo en el método principal. Para la suma se sigue la siguiente regla:

$$(x, y, z) + (a, b, c) = (x + a, y + b, z + c)$$

Para el producto escarlar, se utiliza un ciclo for para ir recorriendo la operación. Se itera para multiplicar las componentes, y para realizar la adición de todas estas. Para el producto escalar se sigue:

$$(x, y, z) \cdot (a, b, c) = xa + yc + zc$$

entonces la primera iteración se multiplican las compornentes vector1[0]*vector2[0], en la segunda las componentes de al posición 1 y en la tercera las componentes de la posición 2. Luego se suman todas ellas, con lo cual debe inicializarse una variable resultado=0, la irá guardadno y sumando las operaciones de la itración. La operación tendrá forma de

Resultado=0
iterar hasta i=2
Resultado=resultado+vector1[i]*vector2[i]

Para el producto vectorial, se siguen la regla del determinante. Sin embargo, en este programa, únicamente se calcularán manualmente los valores de las componentes x,y,z. Este valor tendrá forma de vector, por lo que se opta por utilizar una función sin retorno que imprima el valor y dicha función sea llamada en la función principal.
La operación seguida es:

$$\hat{x} = yc - xb = vector1[1] * vector2[2] - vector1[0] * vector2[1]$$

$$\hat{y} = za - xc = vector1[2] * vector2[0] - vector1[0] * vector2[2]$$

$$\hat{z} = xb - ya = vector1[0] * vector2[1] - vector1[1] * vector2[0]$$

Este valor se imprime en la misma función como [x,y,z].

Entradas y salidas

Se necesita que el usuario ingrese dos vectores en 3 dimensiones. Es decir, vectores de la forma (x,y,z), de los cuales se calculará la magnitud, su suma, su producto escalar y producto vectorial.
→ 2 vectores de la forma (x,y,z)
← magnitud de los vectores, suma, producto escalar, producto vectorial

Código del programa

El código puede ser encontrado como:

```
Usuario: CindyGat, Repositorio LabSimu2021 laboratorio4, problema3.c
```

link:

`https://github.com/CINDYGAT/LabSimu1S2021/blob/main/Laboratorio4/Problema3.c`

Diagrama de flujo

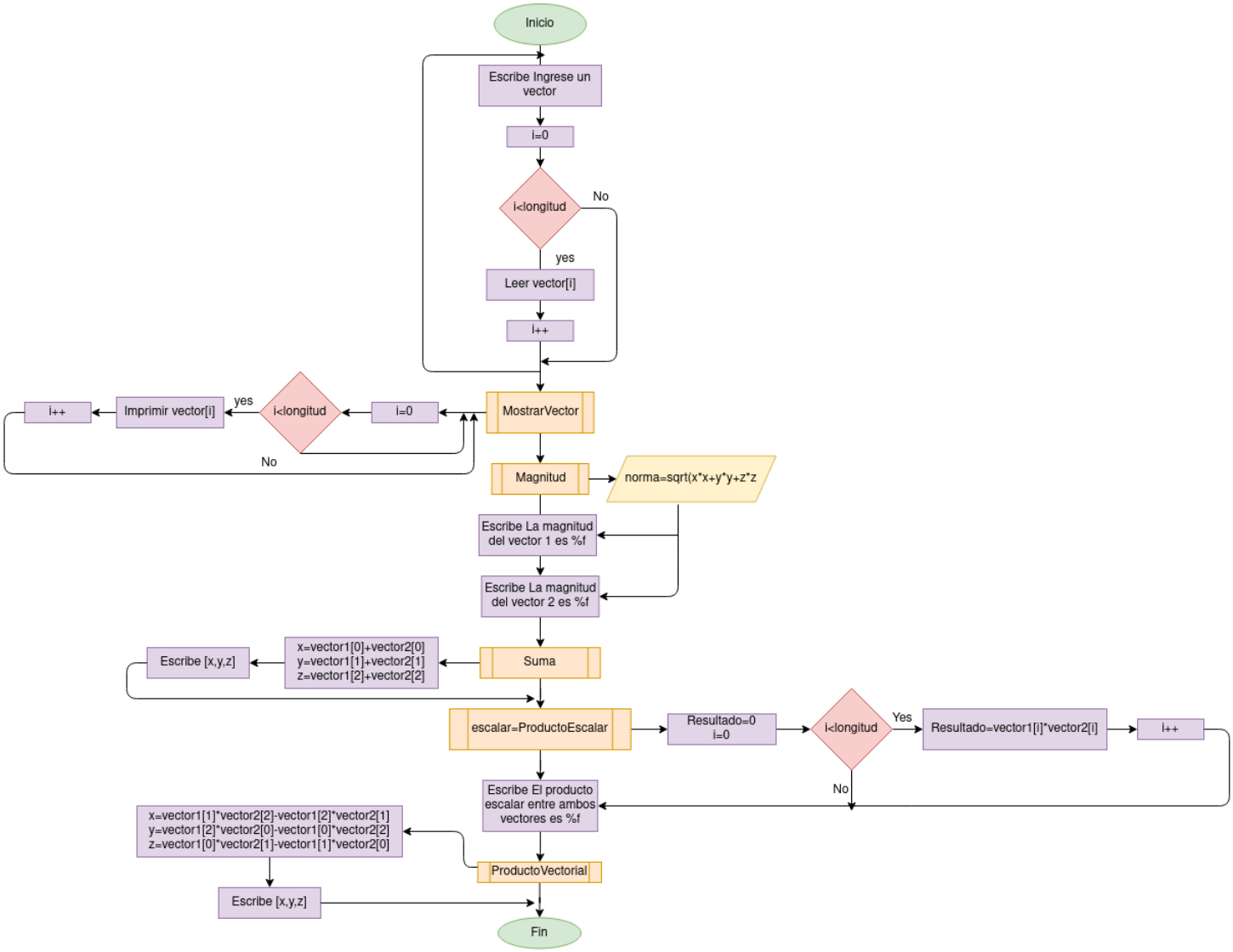


Figura 1: Diagrama de flujo problema 3

Problema 4

Enunciado del problema

Crear un programa que solicite al usuario dos matrices de 3X3 almacenarlas como (matA, matB) y una constante, el programa automaticamente debe de mostrar las los siguientes resultados:

1. matA por constante
2. suma de las dos matrices
3. resta de las dos matrices
4. multiplicacion de las dos matrices
5. determinante de matA
6. transpuesta de matB
7. inversa de matA
8. reduccion de Gauss de matA
9. reduccion de Gauss Jordan matB

Metodología para resolver el problema

El problema consiste en resilver una gran cantidad de operaciones matemáticas. Para ello, se utilizaron varias funciones matemáticas, exactamente 10 de ellas. Las cuales cumplían con mostrar el vector, calcular la multiplicacion por escalar, suma de matrices, resta de matrices, producto, etc.
Para la multiplicacion por escalar, se multiplicaron todas las componentes de la misma por el escalar, es decir:

$$vector[c*i][c*j]$$

Para lograr esto se utilizaron ciclos for, los cuales recorrian toda la matriz.
Para la suma y resta de matrices, se sumaron y restaron sus componentes. de tal forma que

$$vector1[i][j] + vector2[i][j]$$

Para el producto de las matrices, se multiplicaron la fila 1 con la columna 1, la fila 2 con la columna 2, la fila 1 con la columna 2, etc.
Ya que toda multiplicacion por matrices, se tiene que la multiplicacion de todas sus componentes deben sumarse, entonces se utilizó una variable que iba acumulando los valores.

$$sum+ = matrizA[i][g] * matrizB[g][j]$$

Al final, se obtuvo una matriz de 3x3, como se esperaba. Para el determinante de la matriz, se utilizaron las filas como pivote, es decir a11, a21,a31 era los pivotes. Por ello, se iteró

alrededor de las filas.
Se utilizó el principio de los cofactores. De tal forma que las componentes se debían multiplicar y luego sumar/restar. Siguiendo la siguiente estructura:

$$det = det + (matrizA[0][i] * (matrizA[1][(i+1)3] * matrizA[2][(i+2)3] -$$

$$matrizA[1][(i+2)3] * matrizA[2][(i+1)3]));$$

De donde det iba acumulando los valores al cambiar el pivote. En este caso, i=0 hasta 2.
Para encontrar la transpuesta, unicamente se recorrió el la matriz como es usual y se cambió el orden de i y j. Es decir, pasa de ser matrizB[i][j] → matrizB[j][i]
Para encontrar la inversa, se utilizó la forma de:

$$I = \frac{Adj(A)}{|A|}$$

Donde la Adjunta hace referencia a la matriz de cofactores transpuesta.
Nótese que si el determinante es cero, no existe transpuesta.
Para calcular Gauss Jorda se siguieron los siguientes pasos:

1. Se consideró la matriz A ingresada por el usuario como la matriz aumentada.
2. Se intercabian filas cualquiera de ser necesario
3. Recorriendo las columnas primero, verificamos si algun elemento de la c1 era distinto de 1, de no serlo, se dividia dentro del valor.
4. Se tomaban la f1 y f2 considerando las primeras componentes, si estas tenian ceros en los primeros valores, se dejaba como estaba, sino se multiplicaban por el escalar mas conveniente y se restaban.
5. Una vez con ceros debajo de la diagonal, se dividian los valores de la diagonal por escalares convenientes, de tal forma que se obtuviese 1 en la diagonal.

Entradas y salidas

El programa debe ser capaz de calcular diferentes operaciones de matrices, para ello, se ingresan dos matrices de 3x3. Se retornan operaciones matemáticas
→ matrices de 3x3, constante real
← multiplicación por constante, resta de matrices, suma de matrices, multiplicación de matrices, determinante, transpuesta, inversa, reducción de Gauss, Reducción de Gauss Jordan.

Diagrama de flujo

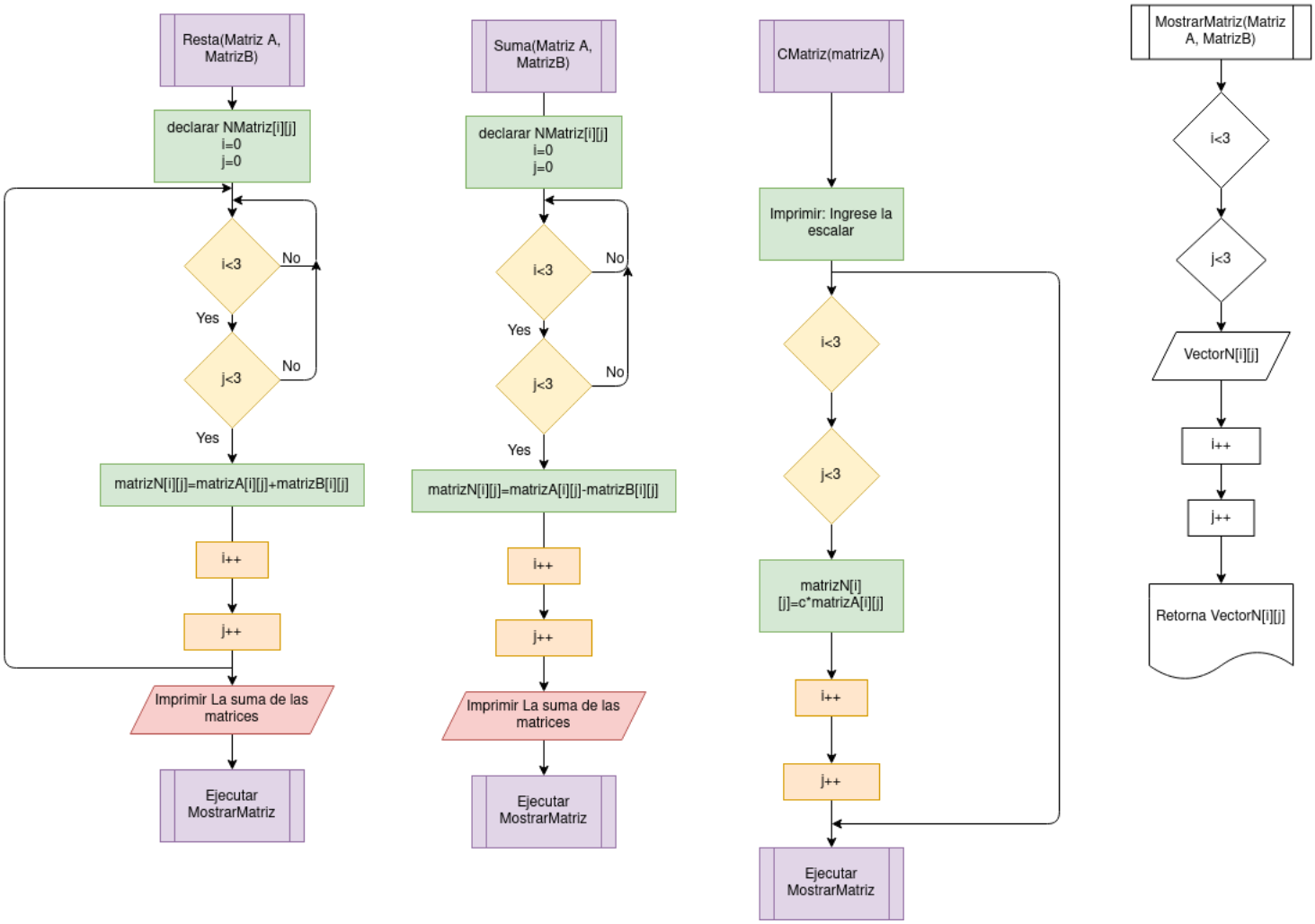


Figura 2: Diagramas de flujo, problema 4

Código del programa

El código puede ser encontrado como:

`Usuario: CindyGat, Repositorio LabSimu2021 laboratorio4, problema4.c`

link:

`https://github.com/CINDYGAT/LabSimu1S2021/blob/main/Laboratorio4/Problema4.c`

Problema 6

Enunciado del problema

Crear un programa que realice la sumatoria desde 1 hasta un numero n que ingrese el usuario de las siguientes funciones.

$$\sum k^2(k-3) \tag{1}$$

$$\sum \frac{3}{k-1} \tag{2}$$

$$\sum 0{,}1(3*2^{k-2}+4) \tag{3}$$

$$\sum \frac{1}{\sqrt{5}}\left[\frac{1+\sqrt{5}}{2}\right]^n - \frac{1}{\sqrt{5}}\left[\frac{1-\sqrt{5}}{2}\right]^n \tag{4}$$

Metodología para resolver el problema

Para el primer problema, únicamente se reescribió el problema tal y como fue dado. Se utilizó la lubreria math.h para que fuese posible utilizar la opcion power.
Para iterar los problemas, se utilizó un ciclo flor, cuyo límite superior era dado por el usuario. Cabe destacar, se se debe pedir al usuario por un número, cuyo número es el valor máximo de iteraciones para todos los programas.

Para el problema dos, primero se notaba que la serie divergia si se consideraba un limite inferior igual a 1, para el resto de los valores, este tendía a algún valor. Por ello, se utiliza un condicional if, tomando en cuenta estas dos opciones.
Se utiliza un ciclo for para iterar entre loslímites establecidos.

Para el problema 3, se realizaron dos funciones que consideraran los fragmentos que tenían potencias a la n. Estas funciones solamente se les deaba el parametro número y devolvuelven el valor de la iteación considerada.

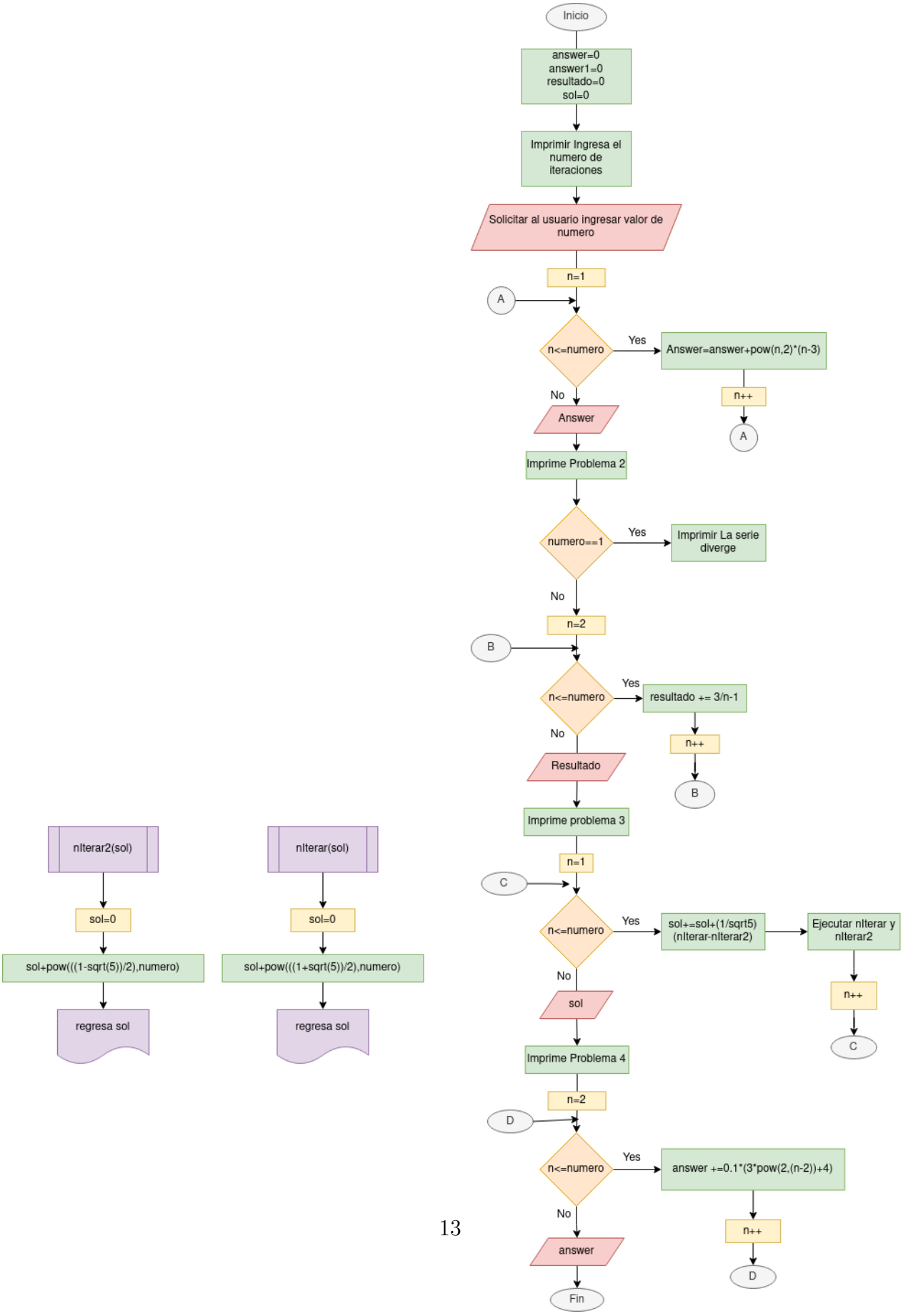
Para el problema 4, se escribe la función tal y como fue indicada. Comenzando la iteración desde 2 hasta el valor dado por el usuario.

Entradas y salidas

Se deben ingresar el numero de iteraciones deseadas. Es decir, hasta donde parará la serie. Como objetos de salida, deben obtenerse el resultado de las series.

$\rightarrow$ numero entero
$\leftarrow$ solución de la serie.

Diagrama de flujo



Código del programa

El código puede ser encontrado como:

```
Usuario: CindyGat, Repositorio LabSimu2021 laboratorio4, problema6.c
```

Link:

https://github.com/CINDYGAT/LabSimu1S2021/blob/main/Laboratorio4/Problema6.c