**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Trabajo Practico N°1

* Garzon, Mario Alfredo, LU: 435

Profesores:

Mg. Ing. Ariel Alejandro Vega

Año: 2024

**REGLAMENTO**

Crear una carpeta denominada TP01\_XXXX donde XXXX es el apellido\_nombre del estudiante. Al producto final, subirlo en su repositorio y compartir el enlace en formulario.

# Sección Expresiones aritméticas y lógicas

Resolver cada ejercicio en un archivo Word y luego programarlo en Processing. En el caso de la programación crear un archivo por ejercicio.

Ejercicio 1: Evaluar (obtener resultado) la siguiente expresión para A = 2 y B = 5

3\* A - 4 \* B / A ^ 2

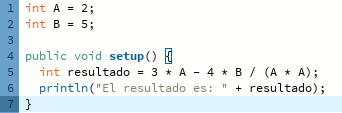
(3\*A)-(4\*B/(A^2))

6-(4\*B/4)

6-5

1

Codificación:



Ejercicio 2: Evaluar la siguiente expresión 4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2

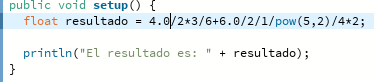
**4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2**

(((4/2) \* 3) /6 ) + ((((6/2) / 1) / (5 ^ 2)) / 4) \* 2

1.0 + 0.06

1.06

Codificación:



Ejercicio 3: Escribir las siguientes expresiones algebraicas como expresiones algorítmicas (en su forma aritmética dentro del algoritmo). En este caso no se pide evaluarlas ni programarlas.

Ejercicio 4: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. **Luego escribirlas como expresiones algebraicas.**

Para aclarar que indicamos con ”Luego escribirlas como expresiones algebraicas” lo aplicamos con el punto a)

𝑏2 − 4. 𝑎. 𝑐

1. b ^ 2 – 4 \* a \* c

|  |  |
| --- | --- |
| Aritmética | Algebraica |
| **b ^ 2 – 4 \* a \* c**  (4 ^ 2) - (4 \* 2 \* 1)  16 – 8  8 |  |

1. 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

|  |  |
| --- | --- |
| Aritmética | Algebraica |
| **3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17**  (3\*(5^4)) - (5 \* (5^3)) + (5\*12) - 17  1875 – 625 + 60 – 17  1293 |  |

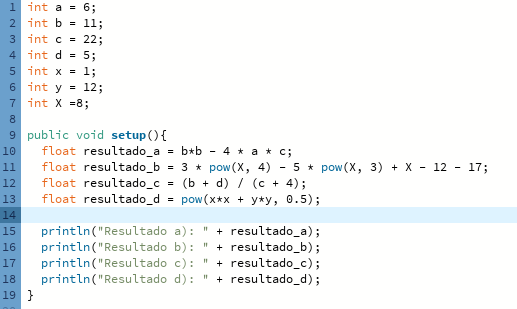
1. (b + d) / (c + 4)

|  |  |
| --- | --- |
| Aritmética | Algebraica |
| **(b + d) / (c + 4)**  (2 + 4) / (1 + 4)  1.2 |  |

1. (x ^ 2 + y ^ 2) ^ (1 / 2)

|  |  |
| --- | --- |
| Aritmética | Algebraica |
| **(x ^ 2 + y ^ 2) ^ (1 / 2)**  (2^2 + 3^2) ^ (1 / 2)  13 ^ (1 / 2)  3.605512 |  |

Codificación:



Ejercicio 5: Si el valor de A es 4, el valor de B es 5 y el valor de C es 1, evaluar las siguientes expresiones:

1. B \* A – B ^ 2 / 4 \* C

5 \* 4 – ((5 ^ 2) / 4) \* 1

20 - 6.25

13.75

1. (A \* B) / 3 ^ 2

(4 \* 5) / 3 ^ 2

2.2 ...

1. (((B + C) / 2 \* A + 10) \* 3 \* B) – 6

(((5 + 1) / 2 \* 4 + 10) \* 3 \* 5) - 6

((6 / 2 \* 4 + 10) \* 3 \* 5) - 6

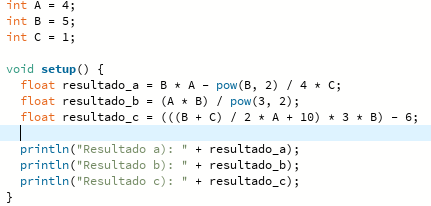
((3 \* 4 + 10) \* 3 \* 5) - 6

(22 \* 3 \* 5) - 6

330 – 6

324

Codificación:



Ejercicio 6: Para x=3, y=4; z=1, evaluar el resultado de

R1 = y+z

R2 = x >= R1

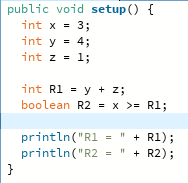
Analisis:

R1= 4+1 = 5

R2= 3 >= R1

Falso

Codificación:



Ejercicio 7: Para contador1=3, contador3=4, evaluar el resultado de

R1 = ++contador1

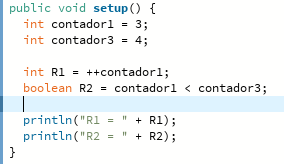
R2 = contador1 < contador2

Análisis:

R2= 4 < 4

R2= falso

Codificación:



Ejercicio 8: Para a=31, b=-1; x=3, y=2, evaluar el resultado de a+b-1 < x\*y

Análisis:

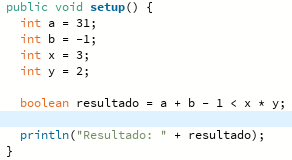
a+b-1 < x\*y

31+(-1)-1 < 3\*2

29 < 12

falso

Codificación:



Ejercicio 9: Para x=6, y=8, evaluar el resultado de

!(x<5)CC !(y>=7)

Análisis:

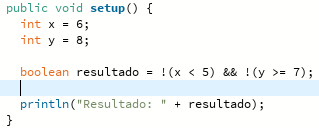
!(x<5) && !(y>=7)

!(6<5) && !(8>=7)

falso && falso

falso

Codificación:



Ejercicio 10: Para i=22, j=3, evaluar el resultado de

!((i>4) || !(j<=6))

Análisis:

!((i>4) || !(j<=6))

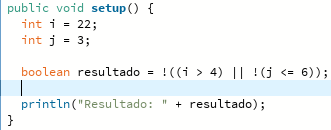
!((22>4) || !(3<=6))

!(verdadero || falso)

!(verdadero)

Falso

Codificación:



Ejercicio 11: Para a=34, b=12,c=8, evaluar el resultado de

!(a+b==c) || (c!=0)CC(b-c>=19)

Análisis:

!(a+b==c) || (c!=0) && (b-c>=19)

!(34+12==8) || (8!=0)&&(12-8>=19)

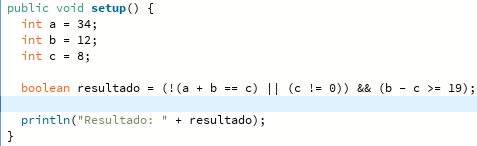
!(46==8) || (8!=0)&&(4>=19)

verdadero || verdadero && falso

verdadero|| falso

verdadero

Codificación:



# Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control

Para cada ejercicio, en el archivo Word agregar las secciones de análisis y diseño, mientras que, para la codificación, crear el archivo de Processing.

Ejercicio 12: Un problema sencillo. Deberá pedir por teclado al usuario un nombre y posteriormente realizará la presentación en pantalla de un saludo con el nombre indicado.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: realizar un saludo personalizado con el nombre ingresado por el teclado.

Análisis:

DATOS DE ENTRADA:

nombre, texto, saludo: string

DATOS DE SALIDA:

Una vez ingresado el nombre, mostrar el saludo en pantalla.

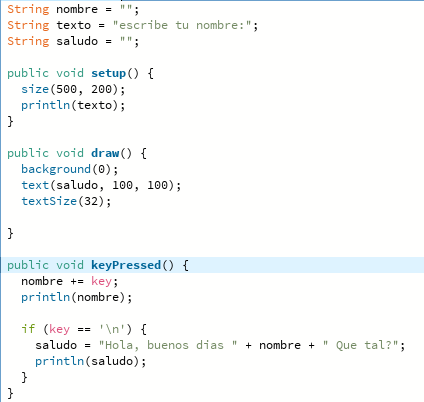
PROCESO:

Ingresar nombre, se toma el nombre y mostrar el saludo en pantalla.

DISEÑO:

|  |
| --- |
| Entidad: Lienzo |
| Variables: nombre, texto, saludo: string |
| Nombre Algoritmo: Saludo  Algoritmo:  Inicio  Nombre, textosaludostring  solicitar el ingreso del nombre por teclado  crear el saludo personalizado  mostrar el saludo personalizado  Fin |

Codificación:



Ejercicio 13: Será común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: calcular el perimetro y el área de un rectángulo.

Análisis:

DATOS DE ENTRADA:

base, altura: float

DATOS DE SALIDA:

Perímetro, área: float

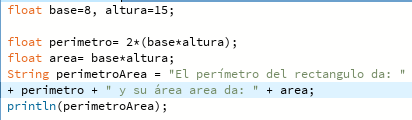
PROCESO:

Calcular el perímetro y el área usando las formulas correspondientes.

DISEÑO:

|  |
| --- |
| Entidad: Algoritmo |
| Variables: base, áltura, perimetro, área, perimetroArea: float |
| Nombre Algoritmo: perimetroAreaRect  Algoritmo:  Inicio  Inicializar base y altura  Establecer las ecuaciones correspondientes  Mostrar los resultados de las ecuaciones  Fin |

Codificación:



Ejercicio 14: Una ayuda importante al momento de resolver problemas con algoritmos es asumir que su gran amigo son las matemáticas. Obtenga la hipotenusa de un triángulo rectángulo conociendo sus catetos



-Desarrollo del punto:

-ANALISIS:

Descripción del problema: obtener la hipotenusa de un triangulo

Análisis:

DATOS DE ENTRADA:

cateto1, cateto2: float

DATOS DE SALIDA:

hipotenusa: float

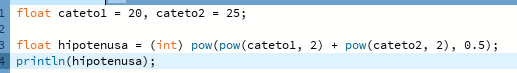
PROCESO:

Calcular la hipotenusa de un triangulo rectángulo usando su formula correspondiente.

DISEÑO:

|  |
| --- |
| Entidad: Algoritmo |
| Variables: cateto1, cateto2, hipotenusa: float |
| Nombre Algoritmo: hipotenusa  Algoritmo:  Inicio  Inicializar cateto1 y cateto 2  Establecer la ecuacion del triangulo rectangulo  Mostrar los resultados de la hipotenusa  Fin |

Codificación:



Ejercicio 15: Si viste algo de los apuntes y vídeos, esto debería ser muy fácil de resolver. Dados dos números permita calcular la suma, resta, multiplicación y división de estos. Considere que cada una de estas operaciones es un algoritmo cuando realice el diseño. Obviamente muestre los resultados.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: resolver las operaciones básicas dado 2 numeros.

Análisis:

DATOS DE ENTRADA:

a, b: entero

DATOS DE SALIDA:

Suma, resta, multiplicación y division

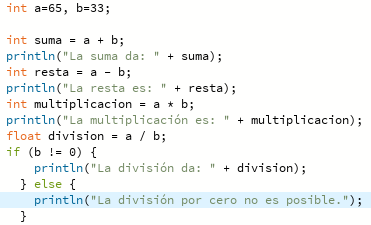
PROCESO:

Calcular la suma,resta, multiplicación y división de los números ingresados.

DISEÑO:

|  |
| --- |
| Entidad: Algoritmo |
| Variables: a, b, suma, resta, multiplicación, división: enteros |
| Nombre Algoritmo: Operaciones Basicas  Algoritmo:  Inicio  Inicializar a y b  Sumar a+b  Restar a-b  Multilicar a\*b  Dividir a/b  Mostrar los resultados de las operaciones  Fin |

Codificación:



Ejercicio 16: Necesitamos convertir una temperatura Fahrenheit en grados Celsius. Si no conoce la forma en la que se realiza esta conversión, debería investigarlo; para eso sirve la etapa de análisis. Pero como somos buenos, daremos una ayuda



-Desarrollo del punto:

-ANALISIS:

Descripción del problema: convertir de Fahrenheit a Celsius.

Análisis:

DATOS DE ENTRADA:

F: float

DATOS DE SALIDA:

C: float

PROCESO:

Conversión de grados Fahrenheit a grados Celsius

DISEÑO:

|  |
| --- |
| Entidad: Algoritmo |
| Variables: F, C: float |
| Nombre Algoritmo: Conversión de grados  Algoritmo:  Inicio  Inicializar F  Conversión del grados dada la ecuacion.  Mostrar la conversion  Fin |

Codificación:



Ejercicio 17: Si queremos representar personajes o power ups (premios) en la pantalla debemos primero ubicarlos en alguna posición dentro de la pantalla. Imagine que está en un juego donde un power up desaparece porque el personaje se acerca a una distancia de x unidades, sin importar por donde se acerque. Por tanto, para que desaparezca, en primer lugar, hay que determinar esa distancia. La forma de representar la posición de un objeto en la pantalla es a través de las coordenadas de un punto. Suponga que la posición de Link está representada por la coordenada (𝑥1, 𝑦1) , mientras que las de la caja de tesoro se halla en la posición (𝑥2, 𝑦2). Si observa con detenimiento se observa la conformación de un triángulo rectángulo, por lo que es posible aplicar Pitágoras para obtener la distancia



Para esto debe calcular el tamaño de los catetos y luego aplicar el teorema. Halle la distancia entre ambos objetos. Cuando programe, represente a lLnk con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: calcular la distancia entre link y el tesoro

Análisis:

DATOS DE ENTRADA:

coorLink, coorTesoro: float

DATOS DE SALIDA:

Distancia entre link y rectangulo

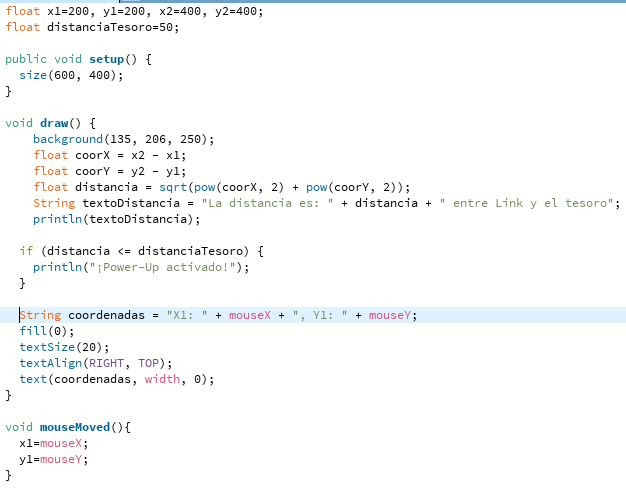
PROCESO:

Calcular la diferencia entre link y el tesosor

DISEÑO:

|  |
| --- |
| Entidad: Algoritmo |
| Variables: x1,y1,x2, y2, coorX, coorY,distancia, disTesoro:float |
| Nombre Algoritmo: distancia  Algoritmo:  Inicio  Inicializar las variables  Darle valores a las coordenadas X e Y  Calcular la distancia  Mostrar la distancia y si el powerUP se activo  Fin |

Codificación:



Ejercicio 18: Desarrolle el análisis y diseño de un algoritmo que permita obtener las raíces de una ecuación de segundo grado. Además, utilice la estructura según para el análisis de la discriminante de la ecuación cuadrática. Obviamente codifique en Processing.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: obtener las raíces de la ecuacion de segundo grado

Análisis:

DATOS DE ENTRADA:

A, b, c: enteros

DATOS DE SALIDA:

Raíces de la ecuacion cuadratica

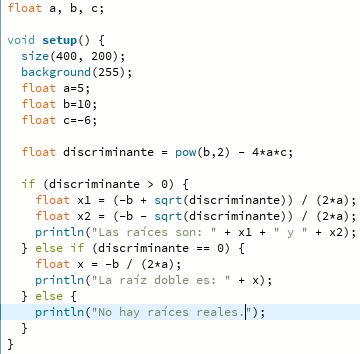
PROCESO:

Calcular el discriminante de ecuacion cuadratica

DISEÑO:

|  |
| --- |
| Entidad: Algoritmo |
| Variables: a, b, c, discriminante:float |
| Nombre Algoritmo: ecuacion cuadratica  Algoritmo:  Inicio  Inicializar las variables a, b, c  Calcular la discriminante  si discriminante > 0 entonces  mostrar ambas raices  si\_no si discriminante = 0 entonces  mostrar la raíz doble  si\_no  mostrar no hay raíces reales  Fin |

Codificación:



Ejercicio 19: Declare las variables necesarias para dibujar una línea que se dibuja desde las coordenadas iniciales del lienzo y se extiende por todo el ancho. Sobre el punto medio de la línea y a una distancia de 40px (en sentido vertical desde la línea) dibuje una elipse que tenga como ancho 80px y de alto 80px. Dentro de la función draw(), actualice las variables necesarias para que la línea desde su inicio se mueva en dirección hacia abajo arrastrando la elipse. Mantenga en cero el valor para background(). Cuando la línea supere la posición de la altura del lienzo, debe invertir su sentido, es decir dirigirse hacia arriba arrastrando la elipse. Cuando la línea alcance nuevamente el valor 0 para su posición en y, el desplazamiento debe ser hacia abajo y así sucesivamente. El lienzo debería verse como en las siguientes figuras



-Desarrollo del punto:

-ANALISIS:

Descripción del problema: obtener las raíces de la ecuacion de segundo grado

Análisis:

DATOS DE ENTRADA:

linea, dir: entero

DATOS DE SALIDA:

Bucle de del circulo y la linea

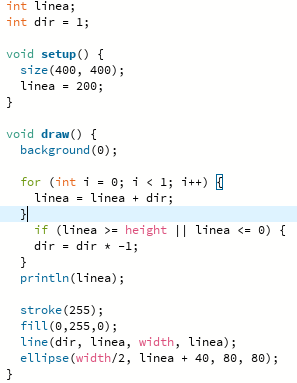
PROCESO:

Dibujar un circulko y una línea que se muevan en bucle de abajo a arriba.

DISEÑO:

|  |
| --- |
| Entidad: Lienzo |
| Variables: linea, dir: entero |
| Nombre Algoritmo: MovimientoBucle  Algoritmo:  Inicio  Inicializar las variables línea y dir  Establecer el ancho y alto en 400  Utilizar la estructura “para” para logra realizar el bucle del movimiento  Mostrar línea  Dibujar línea y circulo  Fin |

Codificación:



Ejercicio 20: Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for. El lienzo debería verse así:



-Desarrollo del punto:

-ANALISIS:

Descripción del problema: Dibujar en el lienzo rectángulos usando estructura interativas.

Analisis:

DATOS DE ENTRADA:

coordenadaRect: coordenadas cartesianas

ancho, alto, distanciaEntreRect, anchoLienzo, altoLienzo:enteros

DATOS DE SALIDA:

Rectangulos dibujados

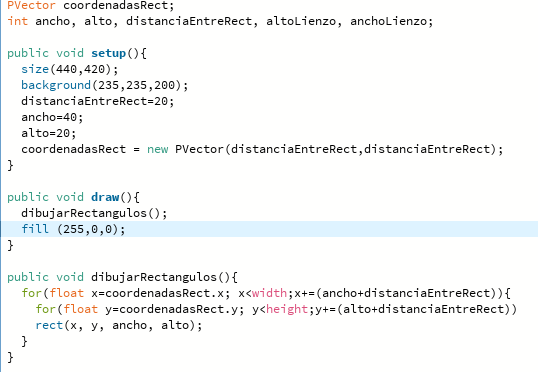
PROCESO:

Dibujar los rectangulos.

DISEÑO:

|  |
| --- |
| Entidad: Lienzo |
| Variables: coordenadaRect: coordenadas cartesianas  ancho, alto, distanciaEntreRect, anchoLienzo, altoLienzo:enteros |
| Nombre Algoritmo: dibujar rectangulos  Algoritmo:  Inicio  anchoLienzo440  altoLienzo420  distanciaEntreRect20  ancho40  alto20  Para x desde coordenadasRect.x hasta anchoLienzo con paso (ancho+distanciaEntreRect) hacer  Para y desde coordenadasRect.y hasta altoLienzo con paso (alto+distanciaEntreRect) hacer  Dibujar un rectángulo en (x, y, ancho, alto)  Fin Para  Fin Para  Fin |

CODIFICACION:



Ejercicio 21: Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto de color rojo



El tamaño del lienzo es size(500,500). La estructura while() se ejecuta dentro de la función setup(). La condición es que solo se dibuje dentro del lienzo. Utilice variables que puedan ayudar a la construcción del dibujo, por ej: x, y, anchoEscalon, altoEscalon, etc.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: Dibujar en el lienzo líneas en forma de escaleras y puntos en cada escalon.

Analisis:

DATOS DE ENTRADA:

pA, pB, pC, pD, distancia: int

DATOS DE SALIDA:

Escalera con puntos en los bordes

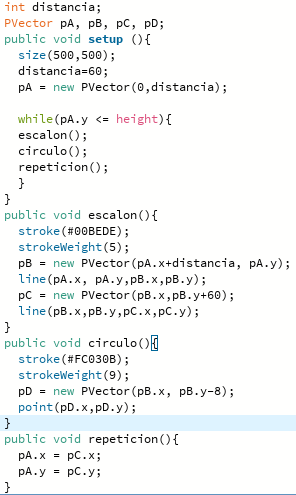
PROCESO:

Dibujar la escalera y los puntos.

DISEÑO:

|  |
| --- |
| Entidad: Lienzo |
| Variables: pA, pB, pC, pD, distancia: int |
| Nombre Algoritmo: escalera\_puntos  Algoritmo:  Inicio  anchoLienzo  altoLienzo  distancia  mientras pA sea menor o igual que ancho lienzo hacer  dibujar línea horizontal  dibujar línea vertical  dibujar circulo  fin mientras  Fin |

CODIFICACION:



Ejercicio 22: Utilizando la estructura de control repetitiva do-while. Replique la siguiente imagen



La imagen debe ser construida desde la función setup(). Defina el tamaño del lienzo en size(600,600), verticalmente se divide el lienzo en franjas de igual medida, se deben dibujar los círculos sobre cada línea de por medio es decir en la línea 1 se dibujan círculos con distanciamiento, en la línea 2 no se dibuja y así sucesivamente. Las líneas tienen un color fijo, los círculos asumen colores aleatorios.

-Desarrollo del punto:

-ANALISIS:

Descripción del problema: Dibujar en el lienzo círculos en medio de unas franjas.

Analisis:

DATOS DE ENTRADA:

distCir, linX, linY, cirX, cirY, anchoLienzo, altoLienzo: int:enteros

DATOS DE SALIDA:

Circulos con color random sobre líneas y un distanciamento de una linea de por medio.

PROCESO:

Dibujar Circulos y lineas

DISEÑO:

|  |
| --- |
| Entidad: Lienzo |
| Variables: distCir, linX, linY, cirX, cirY, anchoLienzo, altoLienzo: int:enteros |
| Nombre Algoritmo: línea\_circulos  Algoritmo:  Inicio  anchoLienzo  altoLienzo  linX0  linY100  distCir  cirY  hacer  dibujar linea en (lineaX, lineaY, anchoLienzo, lineaY)  dibujar circulo en circuloX, circuloY, 50, 50)  circuloX ← circuloX + distanciaCirculo\*2  fin\_hacer  mientras(circuloX sea menor que ancholienzo)  LineaY ← lineaY + 100;  circuloY ← circuloY + 200;  fin\_hacer  Fin |

CODIFICACION:

