|  |  |  |
| --- | --- | --- |
| Texto  Descripción generada automáticamente con confianza media | TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERÍA  Universidad Nacional de Jujuy |  |

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

Trabajo Práctico

N.º 1 taller

Jiménez Jesús Gabriel

TUV000772

Desarrollo del punto

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

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 / 25 / 4 \* 2

(2 \* 3 )/ 6 + 6 / 2 / 1 / 25  / 4 \* 2

(6 / 6 )+ 6 / 2 / 1 / 25 / 4 \* 2

1+ (6 / 2 )/ 1 / 25  / 4 \* 2

1+ (3/ 1 )/ 25 / 4 \* 2

1+ (3/ 25) / 4 \* 2

1+ 0,12 / (4 \* 2)

1+ (0,12 /8)

(1+ 0,06)

1,06

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.

a = 2, b = 3, c = -2, d = 7, x = 5, y = 8

a) b ^ 2 – 4 \* a \* c

    (3 ^ 2) – 4 \* a \* c

    9 – (4 \* 2) \* -2

    9 – (8 \* -2)

    9 – (-16)

    9 + 16

    25



b) 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

    3 \* (5^ 4) – 5 \* (5^ 3) + 5\* 12 – 17

    3 \* (625) – 5 \* (5^ 3) + 5\* 12 – 17

    3 \* 625 – 5 \* (125) + 5\* 12 – 17

    (3 \* 625) – (5 \* 125) +( 5 \* 12) – 17

    1875 - (5 \* 125) +( 5 \* 12) – 17

    1875 - 625 +( 5 \* 12) – 17

    (1875 - 625) + 60 – 17

     1250 + (60 - 17)

     1250 + 43

      1293



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

    (3 + 7) / (-2 + 4)

    10 / (-2 + 4)

     10/2

      5



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

     (5 ^ 2 + 8 ^ 2) ^ (1 / 2)

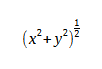
      (25 + 8 ^ 2) ^ (1 / 2)

        (25 + 64) ^ (1 / 2)

         89 ^ (1 / 2)

          89

          9,433981



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:

a) B \* A – B ^ 2 / 4 \* C

    5 \* 4 - (5  ^ 2) / 4 \* 1

    5 \* 4 - 25 / 4 \* 1

    (5 \*4) - 25 / 4 \* 1

    20 - 25 / 4 \* 1

    20 - (25 / 4) \* 1

    20 - (6,25 \* 1)

    20 - 6,25

    13,75

b) (A \* B) / 3 ^ 2

     (4\* 5) / 3 ^ 2

          20  / (3 ^ 2)

       20 / 9

         2,2

c) (((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

     (( 12+ 10) \* 3 \* 5) – 6

     (22 \* 3 \* 5) – 6

     ({22 \* 3} \* 5) - 6

     (66 \* 5) - 6

      330 - 6

324

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

R1 = y+z

R1 = 4 + 1

R1 = 5

R2 = x >= R1

R2 = x >= 5

x no es mayor a 5; la condición es false

Ejercicio 7: Para contador1=3, contador2=4, evaluar el resultado de

R1 = ++contador1

R2 = contador1 < contador2

Resolucion:

R1 = ++3

R1 = 4

R2 = contador1 < contador2

R2 = 4 < 4

R2 = false

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

a+b-1 < x\*y

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

(31+ (-1)) -1 < 6

 30 – 1 < 6

29 < 6

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

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

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

!(false) & !(8 >= 7)

 !(false) & !(true)

false

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

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

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

!((true) || !(3<=6))

!(true || !(true))

!(true || !true)

  !(true || false)

!(true)

false

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

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

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

!(46==8) || (8!=0) & (12-8>=19)

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

!false || (true) & (12-8>=19)

 !false || true & (4>=19)

 !false || true & false

true || true & false

true || false

true

**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.

**Análisis:**

Datos de Entrada:

nombre\_ingresado : Representa el nombre ingresado por el usuario.

Datos de Salida:

mensaje\_saludo : Representa el mensaje de saludo personalizado con el nombre proporcionado.

Proceso:

operador: algoritmo.

Pasos a seguir :

1-ingresar un nombre proporcionado por el usuario.

2-Luego, se genera un saludo que incluye el nombre proporcionado.

3-el programa muestra en pantalla el mensaje de saludo personalizado con el nombre del usuario.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Algoritmo** |
| **Variables:**  **nombre\_ingresado: string**  **mensaje\_saludo: string** |
| **Nombre del Algoritmo:** saludo\_nombre |
| **Proceso del algoritmo:**   1. *Introducir nombre* 2. *Leer nombre\_ingresado* 3. *mensaje\_saludo ← “Genial gusto en conocerte” + nombre\_ingresado* 4. *Mostrar saludo* |

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

**Análisis:**

Datos de Entrada:

base : Representa la base del rectángulo ingresada por el usuario (tipo float).

Altura: Representa la altura del rectángulo ingresada por el usuario (tipo float).

Datos de Salida:

perímetro\_rectángulo: Representa el perímetro del rectángulo calculado (tipo float)

área\_rectángulo: Representa el área del rectángulo calculada (tipo float).

Proceso:

operador: algoritmo.

Pasos a seguir :

1-Ingresar la base del rectángulo proporcionada por el usuario.

2-Ingresar la altura del rectángulo proporcionada por el usuario.

3-Calcular el perímetro del rectángulo utilizando la fórmula: 2 \* (Base + Altura)

4-Calcular el área del rectángulo utilizando la fórmula: Base \* Altura

5-Mostrar perímetro y área por pantalla

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Algoritmo** |
| **Variables:**  **base: float**  **altura: float**  **base\_ingresada: string**  **altura\_ingresada: string**  **perímetro: float**  **área:float** |
| **Nombre del Algoritmo:** Saludo\_nombre |
| **Proceso del algoritmo:**   1. *Leer base* 2. *Leer area* 3. *perimetro ← 2\*(base + altura)* 4. *area ← base \* altura* 5. *perimetro y Area ← “perimetro : ” + perímetro + “ y área : ” + area* 6. *Mostrar perimetro y area por pantalla* |

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

**Análisis:**

Datos de Entrada:

cateto1

cateto2

Datos de Salida:

la hipotenusa del triangulo

Proceso:

operador: algoritmo.

Pasos a seguir :

1-Ingresan los catetos 1 y 2

2- Se calcula la hipotenusa mediante el teorema de pitagoras que es h ² = b² + a²

3- Se despeja la hipotenusa para h = raíz cuadrada de (b² + a² )

5-Mostrar hipotenusa

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Algoritmo** |
| **Variables:**  **cateto1: float**  **cateto2: float**  **cateto1\_ingresado : string**  **cateto2\_ingresado: string**  **hipotenusa: float** |
| **Nombre del Algoritmo:** Calculo\_hipotenusa |
| **Proceso del algoritmo:**   1. *Indicar cateto1 y cateto2* 2. *Leer cateto1* 3. *Leer cateto2* 4. *Calcular hipotenusa que es igual a la raíz de (cateto1^2 + cateto2^2 )* 5. *Mostrar hipotenusa* |

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.

**Análisis:**

Datos de Entrada:

num1

num2

Datos de Salida:

**suma**

**resta**

**multiplicación**

**división**

mensaje de error en caso de 0 como divisor

Proceso:

operador: algoritmo.

Pasos a seguir :

1. Solicitar al usuario que ingrese el valor de num1 y num2
2. Calcular la **suma** de los dos números: Suma = num1 + num2
3. Calcular la **resta** de los dos números: Resta = num1 - num2
4. Calcular la **multiplicación** de los dos números: Multiplicación} =num1 \* num2
5. Calcular la **división** de los dos números: División = num1\num2
6. SI divisor como 0 indicar error
7. Mostrar todos los resultados de las operaciones

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Computadora** |
| **Variables:**   * **num1: int** * **num2: int** * **suma: int** * **resta: int** * **multiplicacion: int** * **division: int** |

|  |
| --- |
| **Nombre del Algoritmo:** **Calculo\_operaciones** |
| **Proceso del algoritmo:**   1. *Indicar los numeros a operar num1,num2* 2. *Leer num1* 3. *Leer num2* 4. *suma ← num1 + num2* 5. *resta ← num1 – num2* 6. *multiplicacion ← num1 \* num2* 7. *mostrar ← “el resultado de la multiplicación es: “ + multiplicacion* 8. *division ← num1 / num2* 9. *Si num2 es 0 mandar mensaje aclaratorio* 10. *mostrar ← “suma es: “ + suma* 11. *mostrar ← “resta es: “ + resta* 12. *mostrar ← “multiplicación es: “ + multiplicacion* 13. *mostrar ← “división es: “ + division* 14. *mostrar ← “la division por cero no se define”* |

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

**Análisis:**

Datos de Entrada: Temperatura en grados Fahrenheit

Datos de Salida: Temperatura en grados Celsius

Proceso:

Operador: Calculadora

Proceso:

1.Ingresar grados Fahrenheit

2.restar 32 a la temperatura en Fahrenheit

3. multiplicar el resultado por 5/9

4. obtener así la temperatura en Celsius.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Computadora** |
| **Variables:**   * **tempFahrenheit: float** * **tempCelsius: float** |

|  |
| --- |
| **Nombre del Algoritmo: Conversión** |
| **Proceso del algoritmo:**   1. *Indicar la temperatura para la conversión* 2. *Leer temperaturaFahrenheit* 3. *tempCelsius ← (5/ 9) \* (tempFahrenheit – 32)* 4. *mostrar temperaturaCelsius* |

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 lInk con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

**Análisis:**

Datos de Entrada: Coordenadas de Link, Coordenadas del tesoro

Datos de Salida: Distancia entre Link y tesoro.

Proceso:

**operador: Processing**

**Proceso:**

1.identificamos las coordenadas de x e y

2. aplicamos el teorema de pitagoras d = raiz de (coordenadas de x)^2 + ( coordenadas de Y )²

3.Al coincidir las coordenadas activara el powerup

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Algoritmo** |
| **Variables:**  **(float igual a real)**  **x1: float**  **y1: float**  **x2:float**  **y2:float**  **x:float**  **y:float**  **distanciatesoro: float**  **distancia:float** |
| **Nombre del Algoritmo:** Calcular\_distancia\_y\_activarPowerup |
| **Proceso del algoritmo:**   1. *Leer x1* 2. *Leer y1* 3. *Leer x2* 4. *Leer y2* 5. *distanciaTesoro ← 100* 6. *x ← x2 - x1* 7. *y← y2 – y1* 8. *distancia ← ((x)^2 + (y)^2)^2* 9. *mostrar “distancia: ” + distancia* 10. *Condicional de si (distancia = distanciaTesoro) entonces mandar mensaje* 11. *mostrar “PowerUp activado!!!”* |

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.

**Análisis:**

Datos de Entrada:

a, b, y c de la ecuación (ax^2 + bx + c = 0)

Datos de Salida:

Las raíces calculadas según el tipo de solución

Proceso:

operador: algoritmo.

Pasos a seguir :

Pasos para resolver una ecuación de segundo grado y encontrar sus raíces:

1. Calcular el discriminante: El discriminante (∆) se calcula como b^2 - 4ac. Este valor determina la naturaleza de las raíces de la ecuación.

2. Determina la naturaleza de las raíces:

- Si ∆ > 0, hay dos raíces reales y distintas.

- Si ∆ = 0, hay una raíz real (raíz doble).

- Si ∆ < 0, no hay raíces reales (las raíces son complejas conjugadas).

3. Calcular las raíces:

- Si ∆ > 0, las raíces se calculan utilizando la fórmula x = (-b ± √∆) / (2a).

- Si ∆ = 0, la raíz se calcula utilizando la fórmula x = -b / (2a).

- Si ∆ < 0, las raíces son complejas y se expresan en la forma x = (-b ± √(-∆)) / (2a).

4. Verificar las soluciones: Sustituye las raíces encontradas

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: persona** |
| **Variables:**   * **a : float** * **b : float** * **c : float** * **discriminante: float** |

|  |
| --- |
| **Nombre del Algoritmo:** encontrar\_raices |
| **Proceso del algoritmo:**   1. *Leer a, b, c* 2. *Calcular discriminante: delta = b^2 - 4ac* 3. *Si delta > 0: Calcular x1 y x2* 4. *Mostrar x1 y x2* 5. *Si delta = 0:* 6. *Calcular x* 7. *Mostrar x* 8. *Calcular parte real e imaginaria de x1 y x2* 9. *Mostrar x1 y x2* 10. Si (discriminante == 0) entonces raiz *← -b / (2\*a)* 11. Mostrar “la raíz doble es: “ + raiz 12. si no mostrar “Nan” |

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

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.

**Análisis:**

Datos de Entrada:

* Ancho del lienzo: 440 píxeles
* Alto del lienzo: 420 píxeles
* Ancho del rectángulo: 40 píxeles
* Alto del rectángulo: 20 píxeles
* Separación horizontal entre rectángulos: 20 píxeles
* Separación vertical entre rectángulos: 20 píxeles

Datos de Salida:

* Dibujo en el lienzo de rectángulos de idénticas medidas (40x20 píxeles)
* Manteniendo una distancia de 20 píxeles entre ellos tanto horizontal como verticalmente.

Proceso:

* 1 Configurar el tamaño del lienzo según los datos de entrada.
* 2 Definir las dimensiones del rectángulo y la separación entre ellos.
* 3 Utilizar estructuras iterativas para dibujar los rectángulos en filas y columnas.
* 4 Calcular las coordenadas de cada rectángulo considerando la separación deseada.
* 5 Dibujar cada rectángulo en su posición correspondiente en el lienzo.
* 6 Continuar hasta cubrir toda la extensión del lienzo con los rectángulos.

**Diseño:**

|  |
| --- |
| **Entidad:** Lienzo |
| **Variables:**   * Ancho = 40 * Alto =20 * Separación horizontal y vertical entre rectángulos: 20 píxeles. * Dimensiones del lienzo: (440,420) píxeles. |
| **Nombre del algoritmo:** Dibujar los rectángulos y mostrarlos por pantalla  **Algoritmo:**  1.Se crea un lienzo de tamaño 440x420 píxeles con un fondo gris.  2.Se dibujan rectángulos anaranjados en el lienzo.  3.Los rectángulos tienen dimensiones de 40x20 píxeles.  4.Hay una separación de 20 píxeles entre cada rectángulo tanto horizontal como verticalmente.  5.Los rectángulos se dibujan de izquierda a derecha y de arriba hacia abajo, como llenando una cuadrícula.  6.Se comienza en la esquina superior izquierda del lienzo.  7.Se dibuja el primer rectángulo y se avanza hacia la derecha.  8.Se repite este proceso hasta llegar al borde derecho del lienzo.  9.Luego se desciende a la siguiente fila y se repite el dibujo de rectángulos de izquierda a derecha.  10.Este proceso se repite hasta llenar todo el lienzo con los rectángulos anaranjados. |

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.

**Análisis:**

Datos de Entrada: pA, pB, pC, pD, distancia

Datos de Salida: una imagen que consiste en escalones con puntos de color rojo en los bordes.

Proceso:

**o**perador:mediante el código en Processing.

**P**roceso:ceso consiste en iterar mediante while() para dibujar escalones y puntos rojos en los bordes

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: programa** |
| **Variables:**   * **puntoA,puntoB,puntoC,puntoD: int** * **distancia : int** |

|  |
| --- |
| **Nombre del Algoritmo:** escalones\_puntos |
| **Proceso del algoritmo:**   1. *anchoLienzo ← 500* 2. *altoLienzo ← 500* 3. *distancia ← 60* 4. *mientras (puntoA.y sea menor o igual que anchoLienzo) Hacer* 5. *dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y)* 6. *dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y)* 7. *dibujar circulo en (puntoD.x, puntoD.y)* 8. *puntoA.x ← puntoC.x* 9. *puntoA.y ← puntoC.y* |

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

**Análisis:**

Datos de Entrada: línes y círculos

Datos de Salida: círculos con colores randoms sobre líneas con un color con distanciamiento por medio

**Proceso:**

**operador:** El recuadro se divide en franjas de igual medida, donde se dibujan lines en todas ellas. En cada líne de forma alternada, se dibujan círculos con colores aleatorios, los cuales están espaciados uniformemente a lo largo de la línea.

**Proceso:**

1. **Configuracion inicial:** Se establece el tamaño del lienzo y se inicializan las variables para las posiciones de la línea y el círculo.
2. **Blucle exterior :** Se repite hasta que la posición del círculo sea menor que el ancho del lienzo.
   * Se dibuja una línea horizontal y un círculo en una posición específica.
   * Se actualiza la posición del círculo para el próximo ciclo.

**3.Blucle interior** Dentro del bucle exterior, se repite hasta que la posición de la línea sea menor que la altura del lienzo.

Se mueve la línea y el círculo a nuevas posiciones para dibujar la siguiente fila de figuras.

**4. Fin :**Una vez que se han dibujado todas las filas de figuras, el proceso termina.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: processing** |
| **Variables:**   * **distanciaCiruclo: int** * **lineaX, lineaY, circuloX, circuloX, distanciaCirculo : int** * **anchoLienzo, altoLienzo: int** |

|  |
| --- |
| **Nombre del Algoritmo:** circulos\_ramdom |
| **Proceso del algoritmo:**   1. *anchoLienzo ← 600* 2. *altoLienzo ← 600* 3. *lineaX ← 0* 4. *lineaY ← 100* 5. *distanciaCirculo ← 30;* 6. *circuloY ← 75* 7. *hacer* 8. *circuloX ← distanciaCirculo* 9. *hacer* 10. *dibujar linea en (lineaX, lineaY, anchoLienzo, lineaY)* 11. *dibujar circulo en circuloX, circuloY, 50, 50)* 12. *circuloX ← circuloX + distanciaCirculo\*2* 13. *mientras(circuloX sea menor que ancholienzo)* 14. *LineaY ← lineaY + 100;* 15. *circuloY ← circuloY + 200;* 16. *fin\_hacer* 17. *mientras (lineaY sea menor que altoLienzo)* |

Fuentes bibliográficas

https://www.hrenatoh.net/curso/processing/processing\_benfry.pdf