

Sección Expresiones aritméticas y lógicas

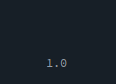
Resolver cada ejercicio en un archivo Word y luego programarlo en Processing. En el caso de la programación crear un archivo por ejercicio.

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 |

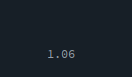
**Captura processing/resultado:**



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

|  |  |
| --- | --- |
| Aritmética | Algebraica |
| 4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2  (((4/2) \* 3) /6) + ((((6/2) / 1) / (5 ^ 2)) / 4) \* 2  ((2 \* 3) /6) + (((3 / 1) / (25))/4) \* 2  1 + (0.12 / 4) \*2  1 + 0.3 \* 2  1 + 0.6  1.06 |  |

**Captura processing/resultado:**



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) b ^ 2 – 4 \* a \* c a= 2, b= 3, c= 5

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| b ^ 2 – 4 \* a \* c  3 ^ 2 – 4 \* 2 \* 5  9 – 40  -31 | B²-4.a.c  3²-4.2.5  9-40  -31 |

**Captura processing/resultado:**



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

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17  3 \* 4 ^ 4 – 5 \* 4 ^ 3 + 4\* 12 – 17  3 \* 256 – 5 \*64 + 48 – 17  768 – 320 + 48 –17  479 |  |

**Captura processing/resultado:**



c) (b + d) / (c + 4) b=3, c= 8, d= 14

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| (b + d) / (c + 4)  (3 + 14) / (8 + 4)  17 / 12  1.41666 |  |

**Captura processing/resultado:**



d) (x ^ 2 + y ^ 2) ^ (1 / 2) x=7, y=3

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| (x ^ 2 + y ^ 2) ^ (1 / 2)  (7 ^ 2 + 3 ^ 2) ^ (1 / 2)  (49 + 9) ^ (1 / 2)  (58) ^ (1 / 2)  7.61577 |  |

**Captura processing/resultado:**



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

a) B \* A – B ^ 2 / 4 \* C

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| B \* A – B ^ 2 / 4 \* C  5 \* 4 – 5 ^ 2 / 4 \* 1  20 – 25 / 4  20 – 6.25  13.75 |  |

**Captura processing/resultado:**



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

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| (A \* B) / 3 ^ 2  (4 \* 5) / 3 ^ 2  20 / 9  2.2222 |  |

**Captura processing/resultado:**



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

|  |  |
| --- | --- |
| **Aritmética** | **Algebraica** |
| (((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) \* 15) -6  ((12 + 10) \*15)-6  (22 \* 15) -6  330 – 6  324 |  |

**Captura processing/resultado:**



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

R1 = y+z

R2 = x >= R1

|  |
| --- |
| R1 = y+z  R2 = x >= R1  false |

**Captura processing/resultado:**



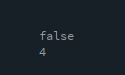
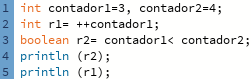
Ejercicio 7: Para contador1=3, contador2=4, evaluar elresultado de

R1 = ++contador1

R2 = contador1 < contador2

|  |
| --- |
| R1 = ++contador1 = 4  R2 = contador1 < contador2 = false |

**Captura processing/resultado:**



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

a+b-1 < x\*y

|  |
| --- |
| a+b-1 < x\*y  31+(-1)-1 < 3\*2  29 < 6  false |

**Captura processing/resultado:**



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

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

|  |
| --- |
| ! (x<5) &&! (y>=7)  ! (6<5) CC! (8>=7)  False && false  false |

**Captura processing/resultado:**



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

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

|  |
| --- |
| !((i>4) || !(j<=6))  !((22>4) || !(3<=6))  !(true || false)  !(true)  false |

**Captura processing/resultado:**



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

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

|  |
| --- |
| !(a+b==c) || (c!=0)&&(b-c>=19)  !(34+12==8) || (8!=0)&&(12-8>=19)  !(46==8) || (8!=0) && (4 >=19)  True || true && false  True || false  True |

**Captura processing/resultado:**



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: text\_nombre // cadena**
* **Datos de salida: text\_mensaje\_saludo // cadena de texto**
* **Proceso:**

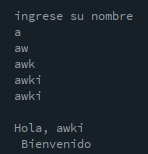
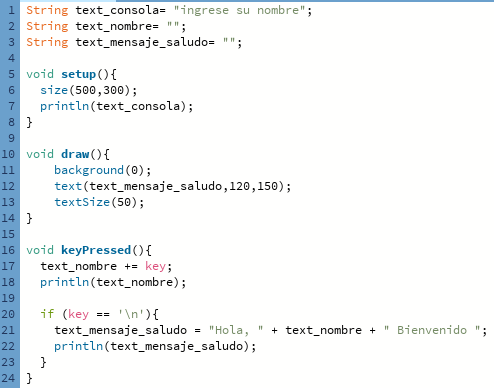
**¿quién realiza el proceso?** El algoritmo o computadora.

**¿qué proceso realiza?** al ingresar un nombre devolverá un saludo con diche nombre mostrándolo en pantalla.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: algoritmo** |
| **Variables:**  **text\_nombre : string //** almacenar nombre  **text\_mensaje\_saludo : string //** almacenar cadena de caracteres |
| **Nombre del algoritmo: mensaje\_saludo** |
| 1. inicio  2. Leer text\_nombre  3. text\_mensaje\_saludo ← “Hola, ” + text\_nombre + “ Bienvenido”  4. Mostrar en pantalla text\_mensaje\_saludo  5. fin |

**Captura processing/resultado:**



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, altura // decimal**
* **Datos de salida: perímetro, área // decimal**
* **Proceso:**

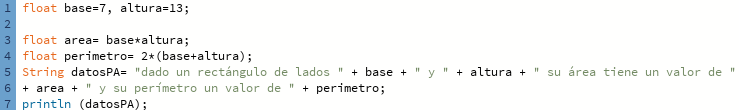
**¿quién realiza el proceso?** el usuario o calculadora

**¿qué proceso realiza?** Calcular el perímetro y el área del rectángulo teniendo en cuenta las fórmulas:

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora** |
| **Variables:**  **Base: float // guardar valor decimal**  **Altura: float // guardar valor decimal**  **Perímetro: float //**  **Área: float //**  **DatosPA: float //** |
| **Nombre del algoritmo: calcular\_perimetro\_area** |
| 1. inicio  2. Leer base  3. leer altura  4. areabase \* altura  5. perímetro 2(base + altura)  6. datosPA“ dado un rectángulo de lados” + base + “y” + altura + “su área tiene un valor de” + área + “y su perímetro un valor de” + perímetro  7. mostrar datosPA  8. fin |

**Captura processing/resultado:**



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: catetoA, catetoB // decima**
* **Datos de salida: hipotenusa // decimal**
* **Proceso:**

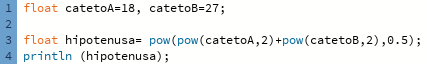
**¿quién realiza el proceso?** usuario o calculadora.

**¿qué proceso realiza?** Calcular la longitud de la hipotenusa del triángulo teniendo en cuenta el valor de sus catetos con la siguiente formula:

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora** |
| **Variables:**  **catetoA: float // guardar valor decimal**  **catetoB: float // guardar valor decimal**  **hipotenusa: float // guardar el valor del calculo** |
| **Nombre del algoritmo: calcular\_hipotenusa** |
| 1. inicio  2. Leer catetoA  3. leer catetoB  4. hipotenusa(catetoA ^ 2 + catetoB ^ 2) ^ 0.5  8. mostrar valor hipotenusa  9. fin |

**Captura processing/resultado:**



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: numeroA, numeroB // decima**
* **Datos de salida: suma, resta, multiplicación, division// decimal**
* **Proceso:**

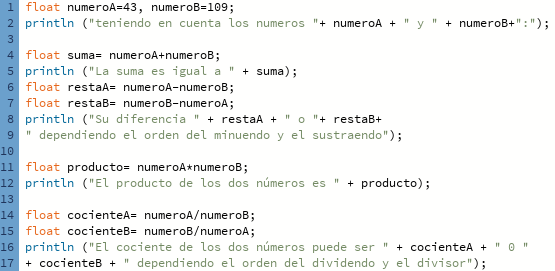
**¿quién realiza el proceso?** calculadora.

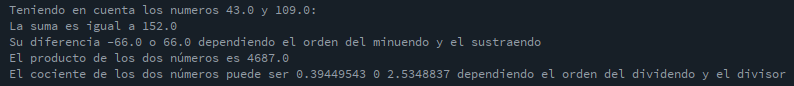
**¿qué proceso realiza?** Calcular las operaciones solicitadas en base a el valor de los números ingresados.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora** |
| **Variables:**  **NumeroA: float // decimal**  **NumeroB: float // decimal**  **Suma: float // decimal**  **Resta: float // decimal**  **Multiplicación: float // decimal**  **División: float // decimal** |
| **Nombre del algoritmo: calcular\_operaciones\_basicas** |
| 1. inicio  2. Leer numeroA  3. leer numeroB  4. mostrar “teniendo en cuenta los numeros ”+ numeroA + “ y ” + numeroB  5. sumanumeroA + numeroB  6. mostrar “su suma es igual a “ + suma  7. restaAnumeroA – numeroB  8. restaBnumeroB – numeroA  9. mostrar “su diferencia “ + restaA + “ o “ + restaB+ “ dependiendo el orden del minuendo y el sustraendo”  10. productonumeroA \* numeroB  11. Mostrar “el producto de los dos números es “ + producto  12. cocienteAnumeroA / numeroB  13. cocienteBnumeroB / numeroA  14. Mostrar “el cociente de los dos números puede ser “ + cocienteA + “ 0 “ + cocienteB + “ dependiendo el orden del dividendo y el divisor”  15. fin |

**Captura processing/resultado:**





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: temp\_fahren // decima**
* **Datos de salida: temp\_celsius// decimal**
* **Proceso:**

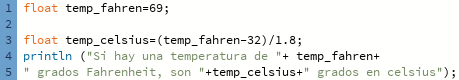
**¿quién realiza el proceso?** Calculadora o algoritmo.

**¿qué proceso realiza?** Convertir una temperatura en grados Fahrenheit a grados celsius atreves de la siguiente fórmula:

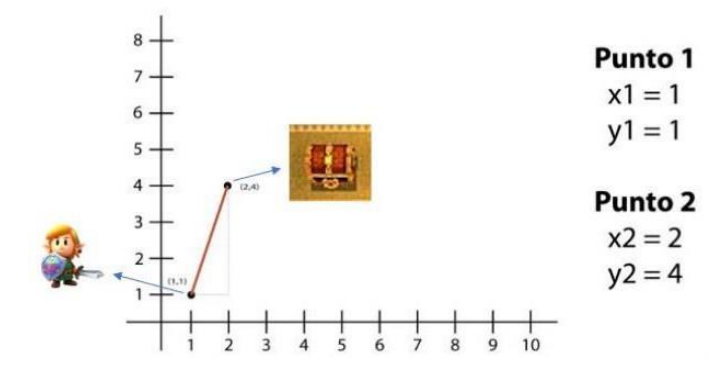
**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **temp\_fahren: float // decimal**  **temp\_celsius: float // decimal** |
| **Nombre del algoritmo: calcular\_operaciones\_basicas** |
| 1. inicio  2. Leer **temp\_fahren**  3. **temp\_celsius(temp\_fahren-32) / 1.8**  **4.mostrar “Si hay una temperatura de “+ temp\_fahren+ “ grados** Fahrenheit, son ”+ **temp\_celsius+” grados en celsius”**  5. fin |

**Captura processing/resultado:**



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.



**Análisis:**

* **Datos de entrada: coordenadas** **link, coordenadas cofre// decima**
* **Datos de salida: distancia entre** **link y cofre// decimal**
* **Proceso:**

**¿quién realiza el proceso?** Calculadora o algoritmo.

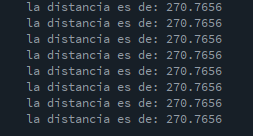
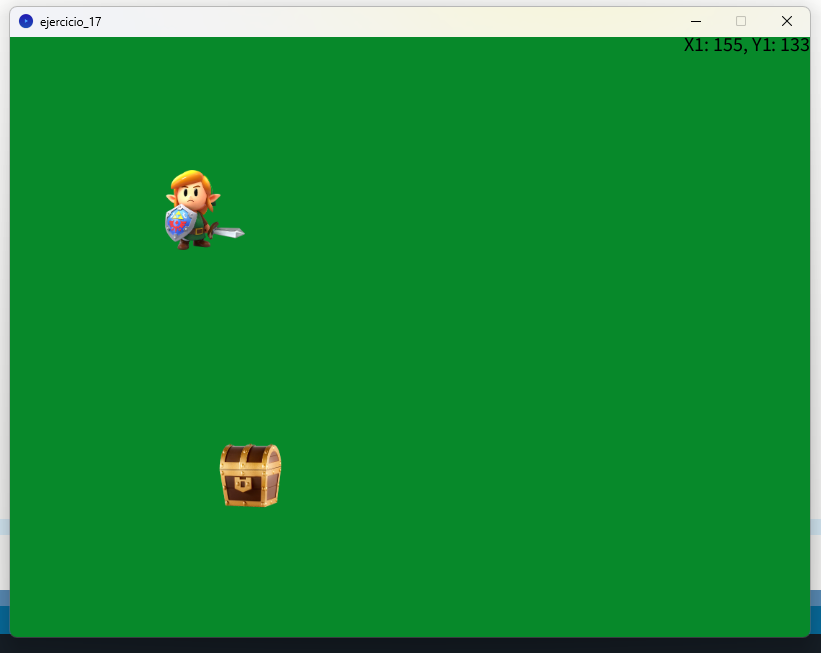
**¿qué proceso realiza?** calcular la distancia entre las variables restando sus coordenadas en X1 y X2 al igual con Y, luego calcular distancia mediante pitadoras, todo esto atreves de la formula:

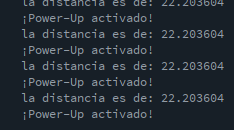
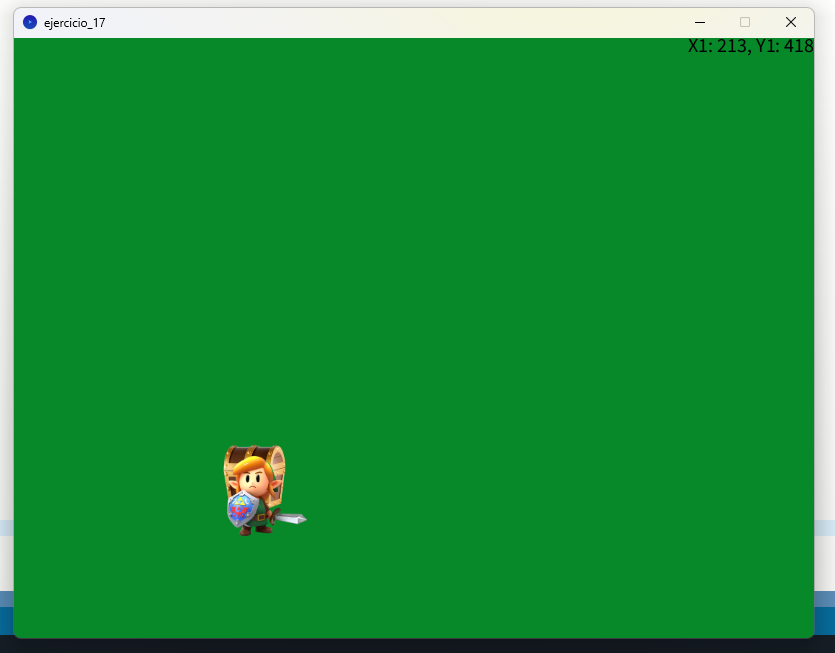
**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **X1: float // decimal**  **X2: float // decimal**  **Y1: float // decimal**  **Y2: float // decimal**  **DistanciaTesoro: float // decimal**  **CoordenadaX: float // almacenar resultado**  **CoordenadaY: float // almacenar resultado**  **Distancia: float // almacenar resultado** |
| **Nombre del algoritmo: calcular\_distancia\_cofre\_link** |
| **Proceso del algoritmo:**   1. **Inicio** 2. **Leer x1** 3. **Leer x2** 4. **Leer y1** 5. **Leer y2** 6. **DistanciaTesoro 50** 7. **coordenadasXx2-x1** 8. **coordenadasYy2-y1** 9. **distancia((coordenadasx)^2+(coordenadasY)^2) ^0.5** 10. **Mostrar “la distancia es de: “+ distancia** 11. **Si (distancia=distanciaTesoro)entonses** 12. **Mostrar “powerUp activado”** 13. **Fin\_si** 14. **Fin** |

**Captura processing/resultado:**







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: valores de la función cuadrática: a, b, c**
* **Datos de salida: raíces de dicha función**
* **Proceso:**

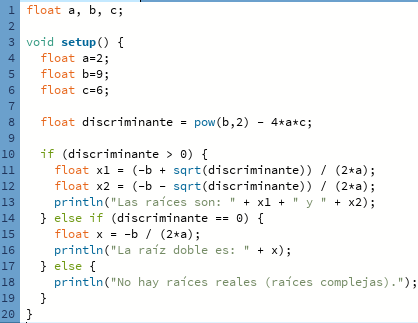
**¿quién realiza el proceso?** Calculadora o algoritmo.

**¿qué proceso realiza?** Calcular las raíces o valores que hacen 0 la ecuación, mediante la formula:

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **A: float // entero**  **B: float // entero**  **C: float // entero**  **Discriminante: float // almacenar resultado**  **Raiz1: float // almacenar resultado**  **Raiz2: float // almacenar resultado** |
| **Nombre del algoritmo: calcular\_raices** |
| **Proceso del algoritmo:**   1. **Inicio** 2. **Leer a** 3. **Leer b** 4. **Leer c** 5. **discriminante b^2 – 4\*a\*c** 6. **si (discriminante > 0) entonce** 7. **raiz1 (-b + (discriminante))^0.5 /(2\*a)** 8. **raiz2 (-b - (discriminante))^0.5 /(2\*a)** 9. **mostrar “las raíces son: ” + raiz1 + “ y ” + raiz2** 10. **si\_no si (discriminante == 0) entonces** 11. **raiz -b / (2\*a)** 12. **mostrar “la raíz doble es: “ + raiz** 13. **si\_no** 14. **mostrar “no hay raíces reales”** 15. **fin** |

**Captura processing/resultado:**



Valores: a=2, b=9, c=6

Valores: a=1, b=-3, c=4

Valores: a=1, b=-2, c=1

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

**Análisis:**

* **Datos de entrada: línea, dir.**
* **Datos de salida: bucle de la línea y elipse**
* **Proceso:**

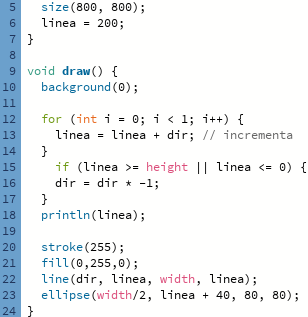
**¿quién realiza el proceso?** Programa **o** algoritmo.

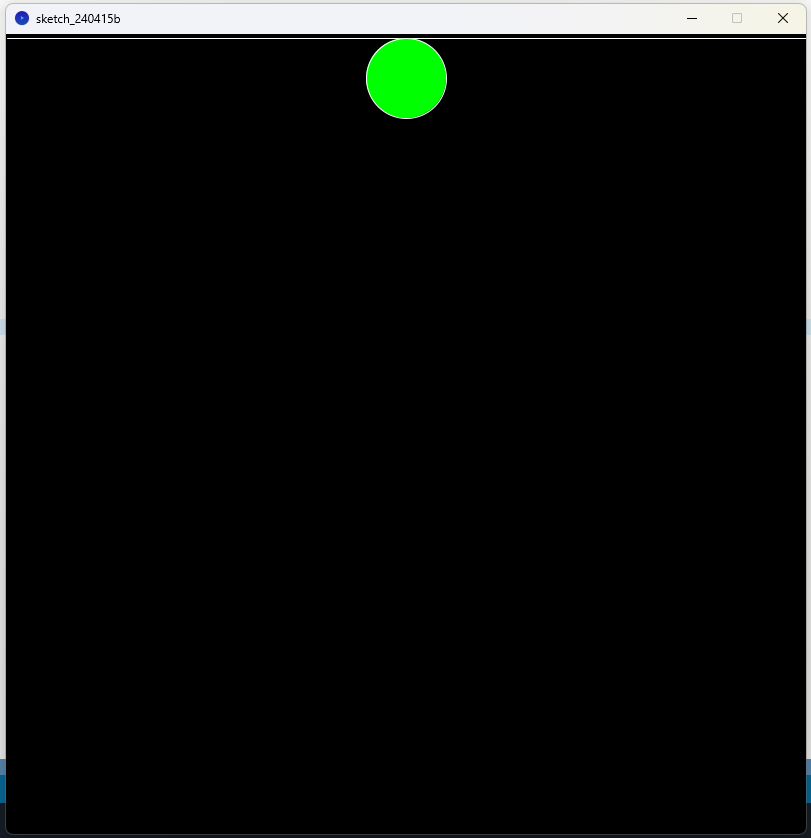
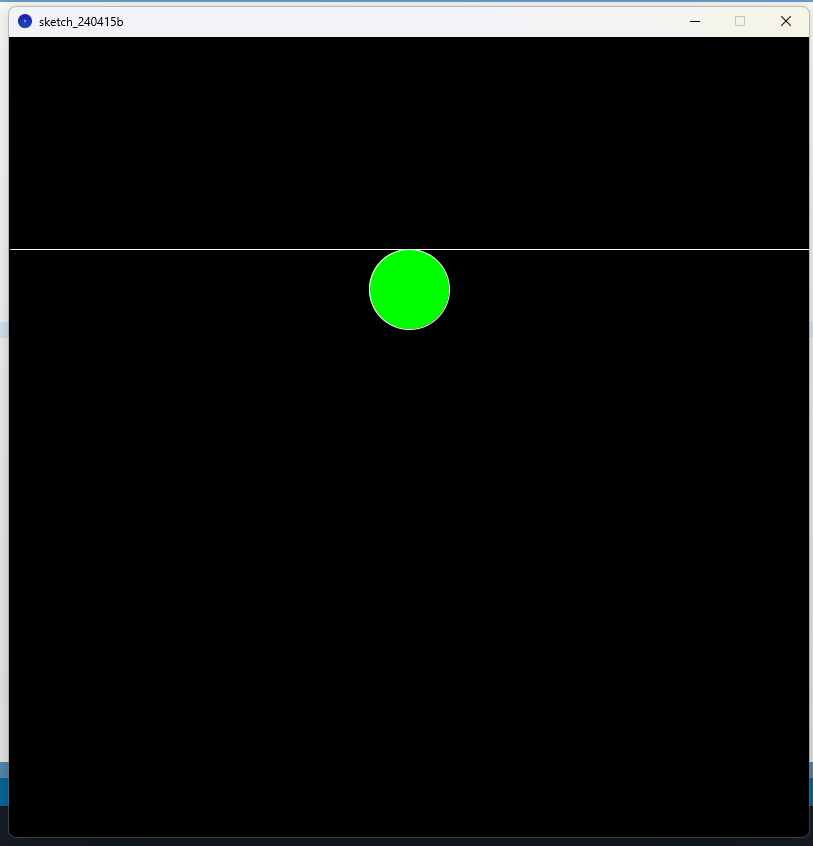
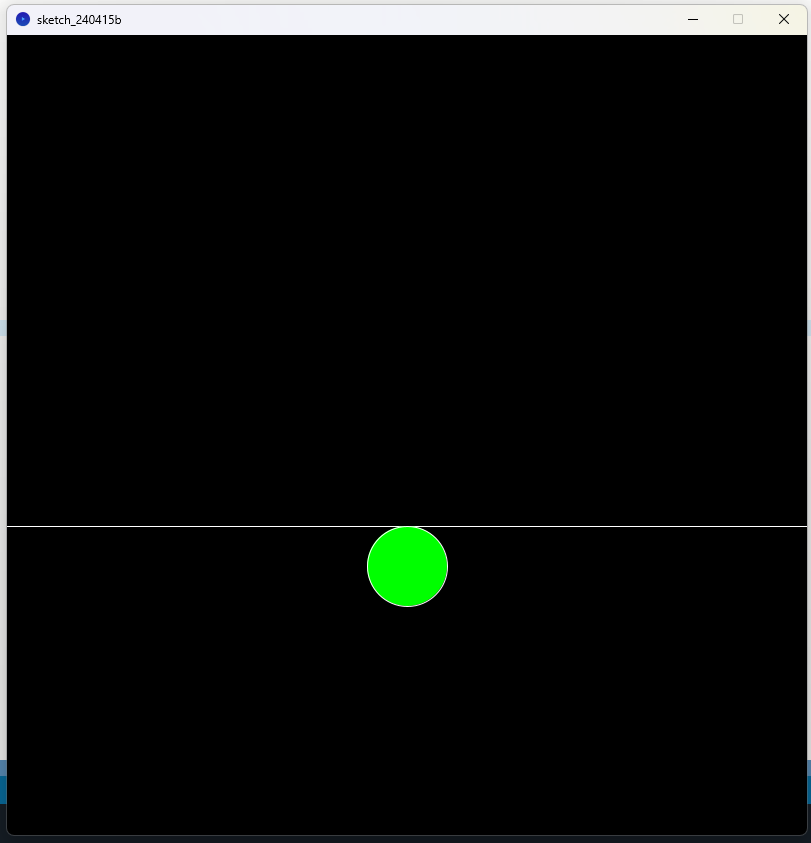
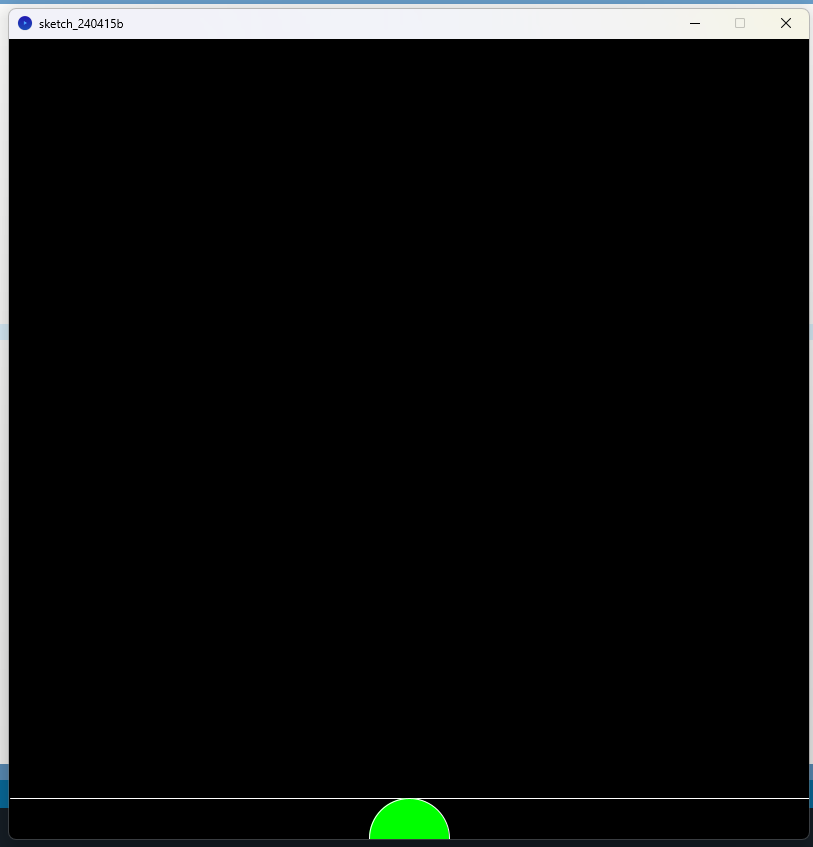
**¿qué proceso realiza?** Generar un bucle en el que la línea y la elipse bajen y suban constantemente.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **Línea: float // entero**  **Dir: float // entero** |
| **Nombre del algoritmo: calcular\_raices** |
| **Proceso del algoritmo:**   1. **Inicio** 2. **Leer linea** 3. **Leer dir** 4. **anchoLienzo ← 800** 5. **altoLienzo ← 800** 6. **para i ← 0 hasta alto incremento 1 hacer** 7. **linea ← linea + dir** 8. **fin\_para** 9. **si ((linea >= anchoLienzo) O (linea <= 0)) entonces** 10. **dir ← dir \* (-1)** 11. **fin\_si** 12. **mostrar linea** 13. **dibujar linea en (dir, linea, altoLienzo, linea)** 14. **dibujar circulo en (altoLienzo/2, linea + 40, 80, 80)** 15. **fin** |

**Captura processing/resultado:**





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:

**Análisis:**

* **Datos de entrada: medidas rectángulo (ancho, alto), distancia entre rectángulo, medidas lienzo**
* **Datos de salida: Dibujo del lienzo con los rectángulos distribuidos correctamente.**
* **Proceso:**

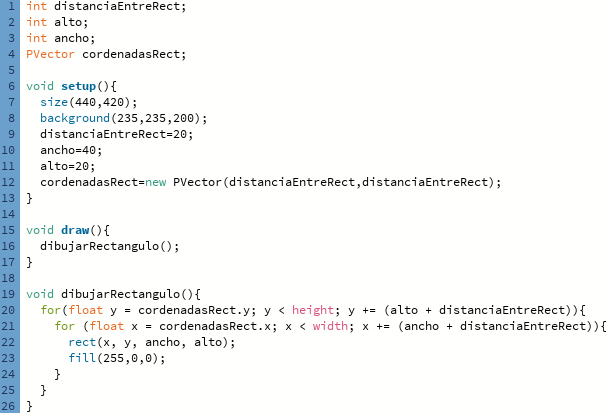
**¿quién realiza el proceso?** Programa o algoritmo.

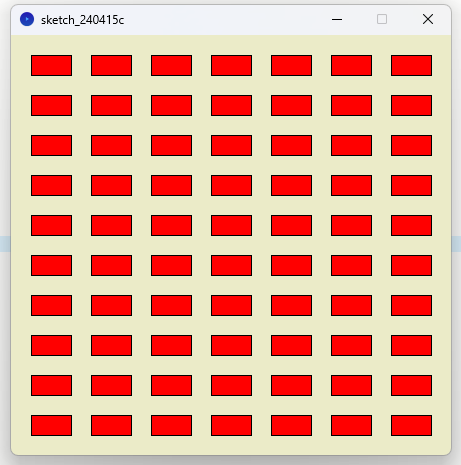
**¿qué proceso realiza?** Dibujar una serie de rectángulos de manera consecutiva en un lienzo separados todos de una distancia especifica

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **coordenadasRect: float // entero**  **ancho, alto, distanciaEntreRect : int // entero**  **anchoLienzo, altoLienzo: int // entero** |
| **Nombre del algoritmo: secuencia \_rectangulos** |
| **Proceso del algoritmo:**   1. **Inicio** 2. **anchoLienzo440** 3. **altoLienzo420** 4. **ancho40** 5. **alto20** 6. **distanciaEntreRect 20** 7. **para x ← coordenadasRect.x hasta anchoLienzo con paso (ancho+distanciaEntreRect)** 8. **Hacer** 9. **para y ← coordenadasRect.y hasta altoLienzo con paso (alto+distanciaEntreRect** 10. **Hacer** 11. **dibujar rectangulo en (x,y,ancho,alto)** 12. **Fin\_para** 13. **Fin\_para** 14. **fin** |

**Captura processing/resultado:**





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:puntoA, puntoB, puntoC, puntoD, distancia**
* **Datos de salida: una imagen que consiste en escalones con puntos de color rojo en los bordes.**
* **Proceso:**

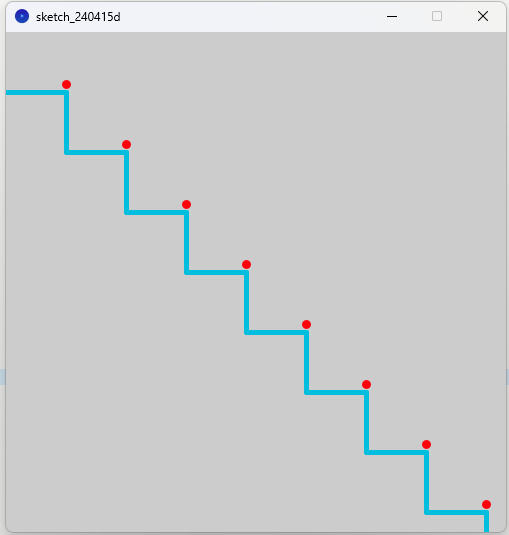
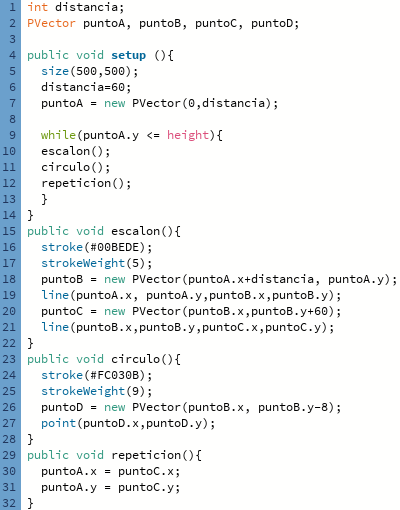
**¿quién realiza el proceso?** Programa o algoritmo.

**¿qué proceso realiza?**  **El proceso consiste en iterar mediante while() para dibujar escalones y puntos rojos en los bordes**

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **puntoA,puntoB,puntoC,puntoD: PVector // almacenar un vector**  **Distancia: int // entero** |
| **Nombre del algoritmo: secuencia \_rectangulos** |
| **Proceso del algoritmo:**   1. **Inicio** 2. **anchoLienzo ← 500** 3. **altoLienzo ← 500** 4. **distancia ← 60** 5. **mientras (puntoA.y sea menor o igual que anchoLienzo) Hacr** 6. **dibujar línea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y)** 7. **dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.)** 8. **dibujar circulo en (puntoD.x, puntoD.y)** 9. **puntoA.x ← puntoC.x** 10. **puntoA.y ← puntoC.y** 11. **fin\_mientras** 12. **fin** |

**Captura processing/resultado:**



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: número de líneas y círculos**
* **Datos de salida: círculos con colores aleatorios sobre líneas con un color con distanciamiento por medio.**
* **Proceso:**

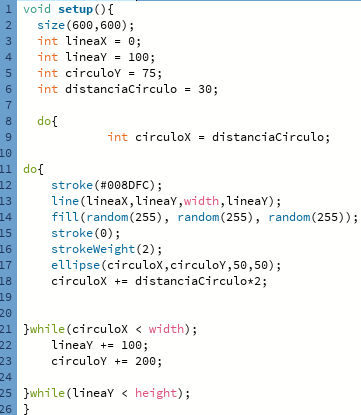
**¿quién realiza el proceso? Programa o algoritmo.**

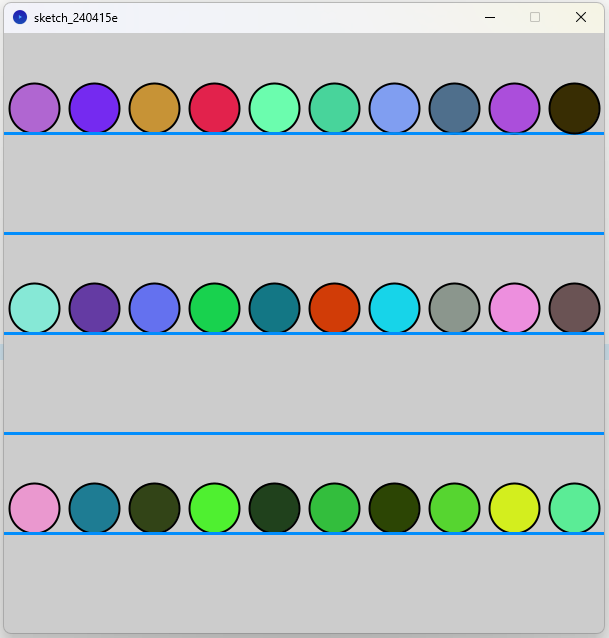
**¿qué proceso realiza? pintar el lanzó con 5 líneas paralelas horizontales y coloradle círculos de colores aleatorios en cada línea de por medio.**

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema: Calculadora o Algoritmo** |
| **Variables:**  **distanciaCiruclo: int //almacena un valor entero**  **lineaX, lineaY, circuloX, circuloX, distanciaCirculo : int //almacena un valor entero**  **anchoLienzo, altoLienzo: int //almacenan valores enteros** |
| **Nombre del algoritmo: secuencia \_rectangulos** |
| **Proceso del algoritmo:**   1. **Inicio** 2. **anchoLienzo ← 600** 3. **altoLienzo ← 600** 4. **lineaX ← 0** 5. **lineaY ← 100** 6. **distanciaCirculo ← 30;** 7. **circuloY ← 75** 8. **Hacer** 9. **circuloX ← distanciaCirculo** 10. **Hacer** 11. **dibujar linea en (lineaX, lineaY, anchoLienzo, lineaY)** 12. **dibujar circulo en circuloX, circuloY, 50, 50)** 13. **circuloX ← circuloX + distanciaCirculo\*2** 14. **fin\_hacer** 15. **mientras(circuloX sea menor que ancholienzo)** 16. **LineaY ← lineaY + 100;** 17. **circuloY ← circuloY + 200;** 18. **fin\_hacer** 19. **mientras (lineaY sea menor que altoLienzo)** 20. **fin** |

**Captura processing/resultado:**





Fuentes:

Manejo de processing:

[PROCESSING: 6.1 PImage [En español] (youtube.com)](https://www.youtube.com/playlist?list=PLtyMmy0eKyqFsLPesmz7y4EznkZFJrGuu)

Referencias de processing:

[Referencia\_Processing\_con\_Imagenes (1).pdf](file:///C:/Users/Usuario/Downloads/Referencia_Processing_con_Imagenes%20(1).pdf)