|  |
| --- |
| **FUNDAMENTOS DE LA PROGRAMACION ORIENTADO A OBJETOS** |

**Trabajo práctico/Actividad**

**N°1**

**Apellido y Nombre – LU/**

**Ceballos Santiago Nicolás**

--------

**Grupo:**

**Integrantes**

**AyN /LU**

**Profesor:**

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

Resolución necesaria en Word:

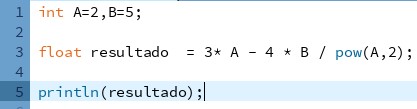
(3\*A)-(4\*B/(A^2))

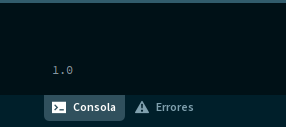
6-(4\*B/4)

6-5

1

Captura de Processing



Resultado:

Ojo: Colocar la captura, no reemplaza que deban agregar a la carpeta el archivo .pde que contiene el código programado.

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

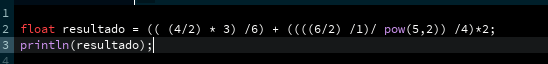
4 / 2\*3 / 6 + 6 / 2 / 1 / 5^2 / 4 \*2

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

1 + 0.06

1.06

Captura de processing:



Resultado:



Ejercicio 3: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. **Luego escribirlas como expresiones algebraicas.**

1. b ^ 2 – 4 \* a \* c

a=2 b=4 c=1

Aritmetrica:

(b^2) – (4 \* a \* c)

16 - 8

8

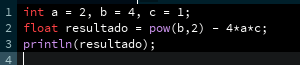
Algebraica:

B2 - 4ac

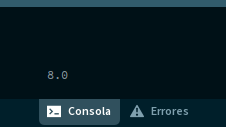
42  - 4.2.1

16 – 8 = 8

Captura de Processing:



Resultado:



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

X=5

Aritmetrica:

3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17

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

1875 – 625 + 60 -17

1293

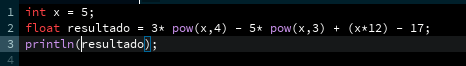
Algebraica:

3x4 – 5x3 + x12 – 17

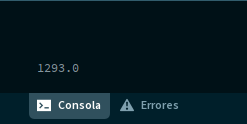
3 . 54 – 5 . 53 + 5 . 12 - 17

1875 – 625 + 60 – 17 = 1293

Captura de Processing:



Resultado:



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

b= 4 c= 8 d= 7

Aritmetrica:

(b+d) / (c+4)

(4+7) / (8+4)

0.91

Algebraica:

= 0.91

Captura del Processing:



Resultado:



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

X = 2 y = 5

Aritmetrica:

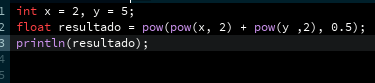
(x ^ 2 + y ^ 2) ^ (1 / 2)

((2^2) + (5^2))^(1/2)

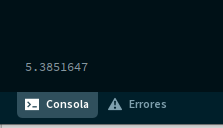
5.38

Algebraica:

Captura del Processing:



Resultado:



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

𝑏2 − 4. 𝑎. 𝑐

Ejercicio 4: 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

Aritmetrica:

B \* A – B ^ 2 / 4 \* C

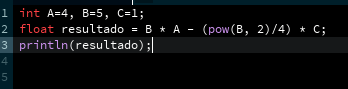
5\*4 – ((5^2) / 4).1

13.75

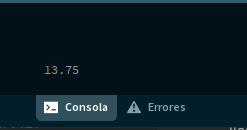
Algebraica:

= 13.75

Captura del Processing:



Resultado:



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

Aritmetrica:

(A \* B) / 3 ^ 2

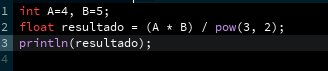
(4 \* 5) / 3 ^ 2

2.22

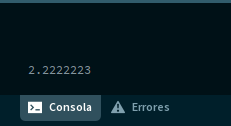
Algebraica:

=2.22

Captura del Processing:



Resultado:



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

Aritmetrica:

(((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

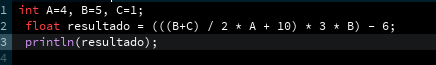
330 – 6

324

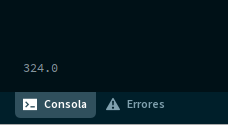
Algebraica:

((

Captura del Processing:



Resultado:



Ejercicio 5: Para x=3, y=4; z=1, evaluar el resultado de

R1 = y+z

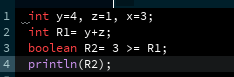
R2 = x >= R1

R1 = 4 + 1 =5

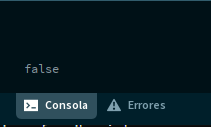
R2 = 3 >= R1

Falso

Captura del Processing:



Resultado:



Ejercicio 6: Para contador1=3, contador2=4, evaluar el resultado de

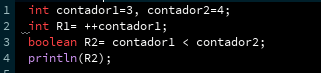
R1 = ++contador1

R2 = contador1 < contador2

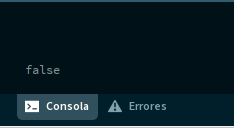
R2= 4 < 4

R2= falso

Captura del Processing:



Resultado:



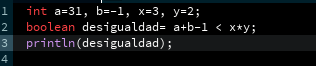
Ejercicio 7: Para a=31, b=-1; x=3, y=2, evaluar el resultado de a+b-1 < x\*y

31 + (-1) – 1 < 3 \* 2

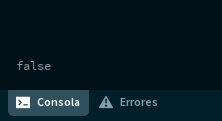
29 < 12

Falso

Captura del Processing:



Resultado:



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

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

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

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

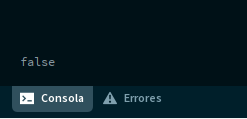
falso = falso

falso

Captura del Processing:



Resultado:



Ejercicio 9: Para i=22, j=3, evaluar el resultado de

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

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

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

¡(verdadero) || falso)

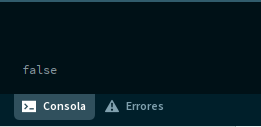
¡(verdadero)

falso

Captura del Processing:



Resultado:



Ejercicio 10: Para a=34, b=12,c=8, evaluar el resultado de

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

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

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

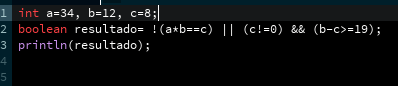
!(46==8) || (8!=0) && (4>=19)

verdadero || verdadero && falso

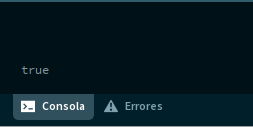
verdadero || falso

verdadero

Captura del 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 11: 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.

Analisis:

Datos de Entrada: nombre\_ingresado

Datos de Salida: mensaje\_saludo

Proceso:

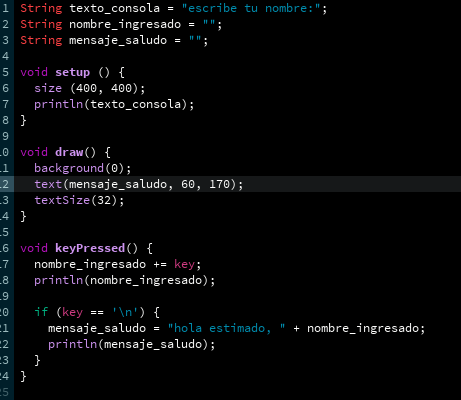
¿Quién Debe realizar el proceso?: El algoritmo

¿Cuál es el proceso que resuelve?: ingresar un nombre que va a devolver un saludo personalizado con el nombre ingresado y su presentación en pantalla.

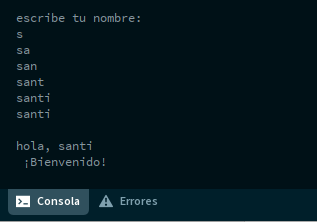
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: Algoritmo |
| Variables:   * nombre\_ingresado // almacena el nombre * mensaje\_saludo // almacena una cantidad de caracteres |
| Nombre del algoritmo:saludar\_nombre  Proceso del algoritmo:   1. Inicio 2. Leer nombre\_ingresado 3. Mensaje\_saludo🡨 “Hola estimado,” + nombre\_ingresado 4. Mostrar saludo 5. fin |

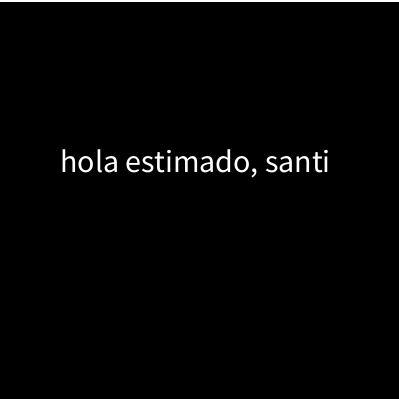
Captura de Processing:



Mensaje en terminal:



Mensaje en lienzo:



Ejercicio 12: Será común resolver problemas utilizando variables. Calcule el perímetro y área de un rectángulo dada su base y su altura.

Analisis:

Datos de Entrada: base, altura

Datos de Salida: perímetro, area

Proceso:

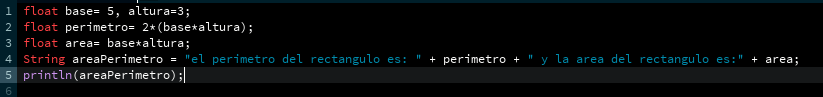
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: calcular el perímetro y área con los datos recibidos, utilizando las formulas correctas.

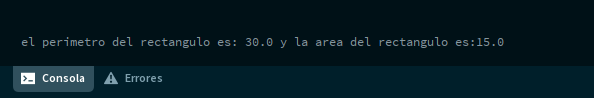
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - base  - altura  - perímetro  - area  - areaPerimetro |
| Nombre del algoritmo:perímetro\_area  Proceso del algoritmo:   1. Inicio 2. Leer base 3. Leer area 4. Perímetro🡨 2\*(base+altura) 5. area🡨 base \* altura 6. areaPerimetro🡨 “el perímetro del rectángulo es: “+ perímetro + “y la área del rectángulo es:” + área 7. mostrar areaPerimetro 8. fin |

Capturas de Processing:



Resultado:



Ejercicio 13: 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



Analisis:

Datos de Entrada: catetoA, catetoB

Datos de Salida: hipotenusa

Proceso:

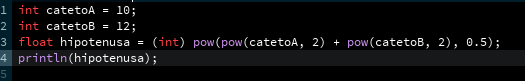
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: se obtiene las longitudes de los catetos y se encuentra la hipotenusa del triangulo rectangulo

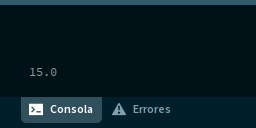
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - catetoA  - catetoB  -hipotenusa |
| Nombre del algoritmo: hipotenusa  Proceso del algoritmo:   1. Inicio 2. Leer catetoA 3. Leer catetoB 4. Hipotenusa🡨(a^2+b^2)^(0.5) 5. Mostrar hipotenusa 6. Fin |

Captura de Processing:



Resultado:



Ejercicio 14: 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.

Analisis:

Datos de Entrada: num1, num2

Datos de Salida: suma, resta, multiplicación, division

Proceso:

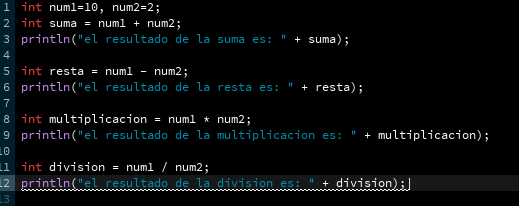
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: se realiza la suma, resta, multiplicación y división de los dos números.

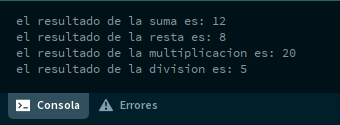
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - num1  - num2 |
| Nombre del algoritmo: calcular  Proceso del algoritmo:   1. Inicio 2. Leer num1 3. Leer num2 4. Suma🡨 num1+num2 5. Mostrar 🡨 ”el resultado de la suma es:” + suma 6. resta 🡨 num1-num2 7. mostrar🡨 “el resultado de la resta es:” + resta 8. multiplicación🡨 num1\*num2 9. mostrar🡨 “el resultado de la multiplicación es:” + multiplicación 10. división🡨 num1/num2 11. mostrar🡨 “el resultado de la división es:” + división 12. fin |

Captura de Processing:



Resultado en la terminal:



Ejercicio 15: 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



Analisis:

Datos de Entrada: temperaturaFahrenheit

Datos de Salida: temperaturaCelsius

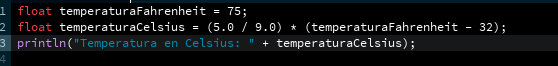
Proceso:

¿Quién Debe realizar el proceso?: el programa

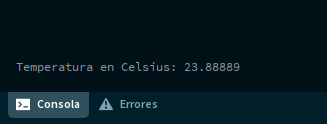
¿Cuál es el proceso que resuelve?: transformar los grados Fahrenheit a grados Celcius

Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - temperaturaFahrenheit  - temperaturaCelsius |
| Nombre del algoritmo: convertir\_temperatura\_Fahrenheit\_Celsius  Proceso del algoritmo:   1. Inicio 2. Leer temperaturaFahrenheit 3. temperaturaCelsius🡨(5.0/9.0)\*(temperaturaFahrenheit – 32) 4. mostrar temperaturaCelsius 5. fin |

Captura de Processing:

Resultado en terminal:



Ejercicio 16: 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.

Analisis:

Datos de Entrada: coordenadas de Link, coordenadas del tesoro

Datos de Salida: distancia entre link y el tesoro

Proceso:

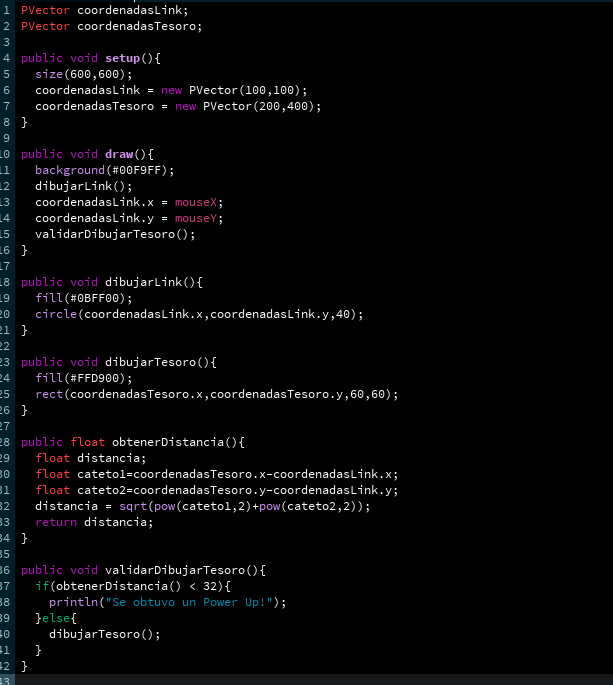
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: calcular la distancia entre link y el tesoro con los catetos que se forman entre ellos.

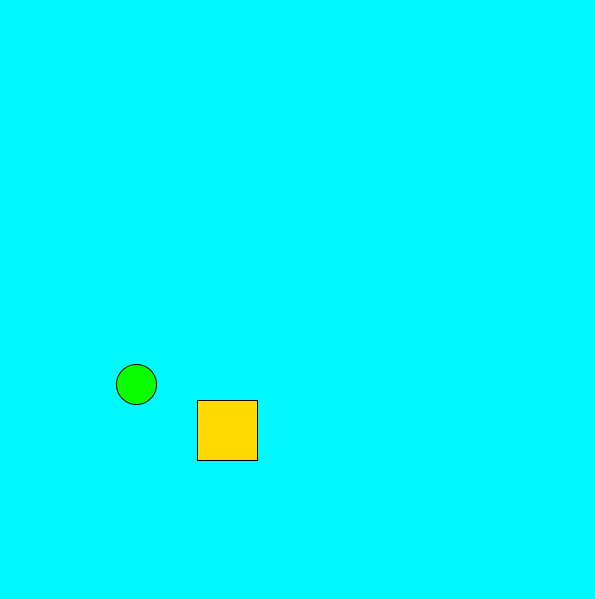
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - coordenadasLinkX  - coordenadasLinkY  - coordenadasTesoroX  - coordenadasTesoroY  - distancia |
| Nombre del algoritmo: distancia\_Link\_Tesoro  Proceso del algoritmo:   1. Inicio 2. Leer coordenadasLinkX 3. Leer coordenadasLinkY 4. Leer coordenadasTesoroX 5. Leer coordenadasTesoroY 6. distancia🡨((cateto1)^2+(cateto2)^2)^2 7. si (distancia🡨coordenadasTesoro) entonces 8. mostrar “Se obtuvo un Power Up!” 9. fin |

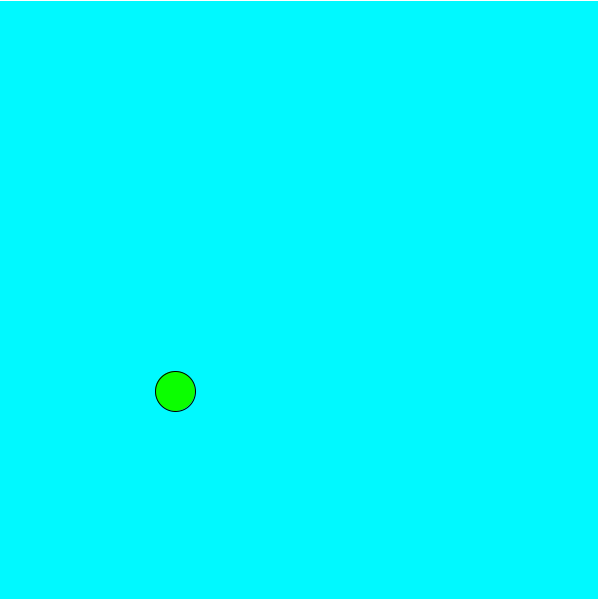
Captura de Processing:



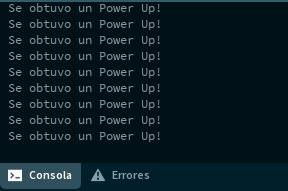
Lo que se muestra en el lienzo:



Cuando se acerca Link al Tesoro:



Mensaje que aparece en la terminal:



Ejercicio 17: 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.

Analisis:

Datos de Entrada: coeficientes de la ecuación cuadrática a, b y c.

Datos de Salida: raíces de la ecuación cuadratica

Proceso:

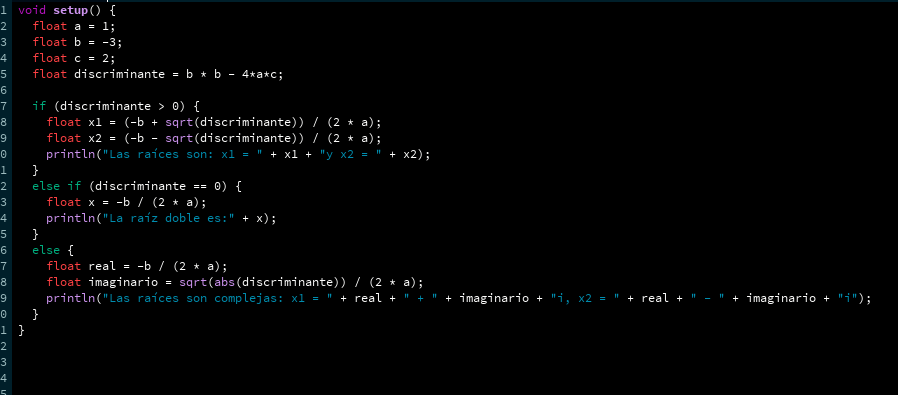
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: calcula la discriminante, luego procede a calcular las raíces.

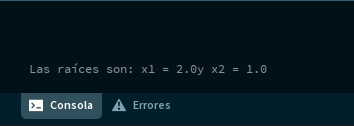
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - a  - b  - c  - discriminante |
| Nombre del algoritmo: buscar\_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) entonces 7. Raiz1 🡨 (-b + (discriminante))^0.5/(2\*a) 8. Raiz2 🡨 (-b – (discriminante))^0.5/(2\*a) 9. Mostrar “Las raíces son: x1 = “ + raíz1 + “ y x2 = “ + raiz2 10. Sino (discriminante == 0) entonces 11. Raíz 🡨 -b/(2\*a) 12. Mostrar “la raíz doble es: “ + raíz 13. Sino 14. real 🡨 -b/(2\*a) 15. imaginario 🡨 (discriminante)/(2\*a) 16. mostrar “las raíces son complejas: x1 = “ + real “ + “ + imaginario + “ i, x2 = “ + real + “ – “ + imaginario + “i” 17. fin |

Captura de Processing:



Resultado:



Ejercicio 18: 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



Analisis:

Datos de Entrada: dir, linea

Datos de Salida: rebote constante de la línea y el circulo.

Proceso:

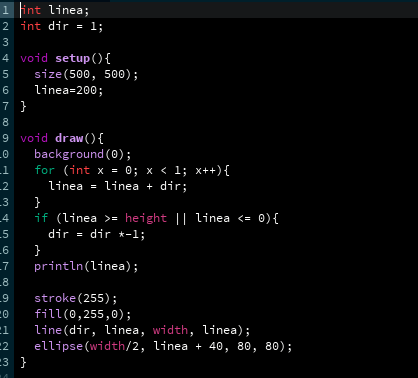
¿Quién Debe realizar el proceso?: el programa

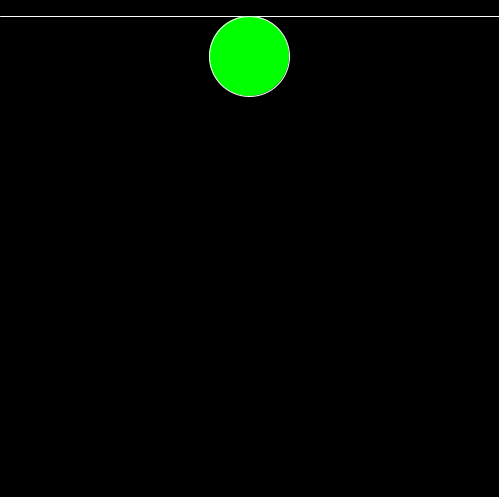
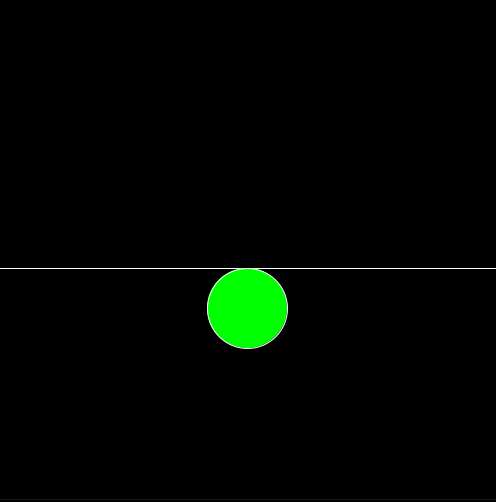
¿Cuál es el proceso que resuelve?: se dibuja la línea y se logra que rebote arriba y abajo del lienzo, luego se le va a colocar un circulo para que lo lleve.

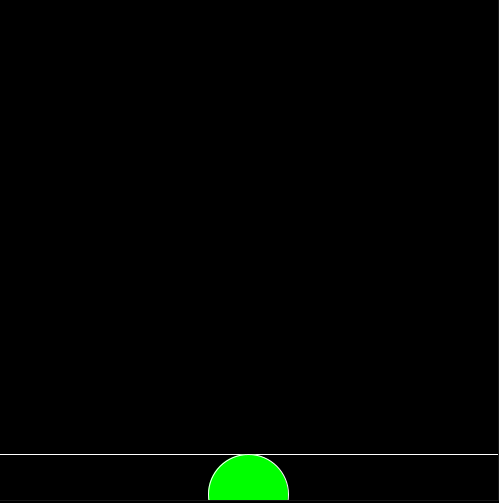
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - linea  - dir |
| Nombre del algoritmo: bucle\_linea\_circulo  Proceso del algoritmo:   1. Inicio 2. Leer línea 3. Leer dir 4. anchoLienzo🡨500 5. altoLienzo🡨500 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 de Processing:



Resultado en Lienzo:



Ejercicio 19: 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:



Analisis:

Datos de Entrada: rectángulos con las especificaciones solicitadas.

Datos de Salida: rectángulos dibujados manteniendo la distancia solicitada.

Proceso:

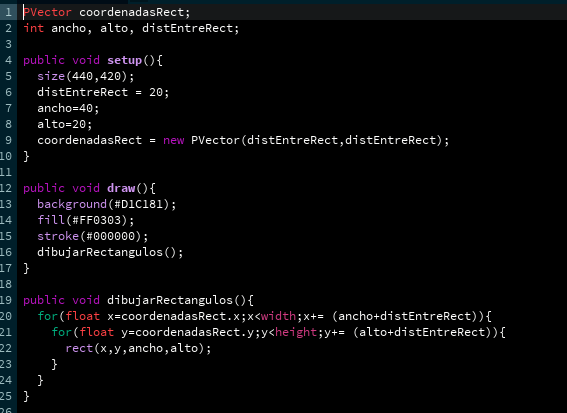
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: se dibuja una serie de rectángulos con los tamaños solicitados, manteniendo una distancia especifica entre ellos tanto horizontal como vertical.

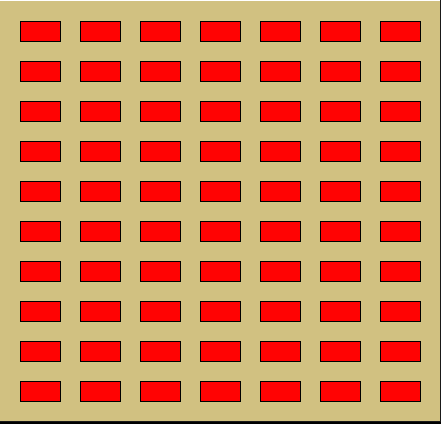
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - coordenadasRect  -ancho  -alto  -distEntreRect  -anchoLienzo  -altoLienzo |
| Nombre del Algoritmo:dibujar\_rectangulos |
| Proceso del algoritmo:   1. Inicio 2. anchoLienzo🡨440 3. altoLienzo🡨420 4. distEntreRect🡨20 5. ancho🡨40 6. alto🡨20 7. para x coordenadasRect.x hasta anchoLienzo con paso (ancho+distEntreRect) 8. para y coordenadasRect.y hasta altoLienzo con paso (alto+distEntreRect) 9. dibujar un rectángulo en (x.coordenadas.y) con dimensiones ancho y alto 10. fin |

Captura de Processing:



Lienzo:



Ejercicio 20: 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.

Analisis:

Datos de Entrada: puntoA, puntoB, puntoC, puntoD, distancia

Datos de Salida: un escalon con círculos rojos en los bordes.

Proceso:

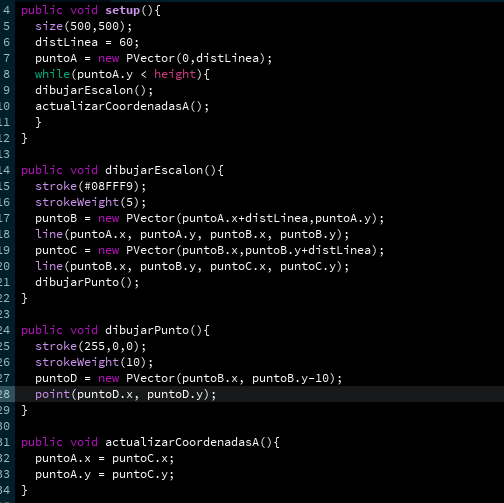
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: usar while para colocar escalones con puntos rojos en el borde

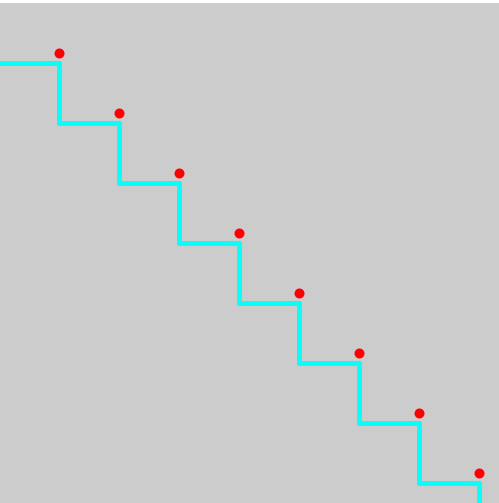
Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - puntoA  - puntoB  - puntoC  - puntoD  - distancia |
| Nombre del escalones\_puntos |
| Proceso del algoritmo:   1. Inicio 2. anchoLienzo🡨500 3. altoLienzo🡨500 4. distancia🡨60 5. mientras (puntoA.y < sea menor o igual que al altoLienzo) 6. dibujar linea horizontal en (puntoA.x, puntoA.y, puntoB.x, puntoB.y) 7. dibujar línea vertical en (puntoB.x, puntoB.y, puntoC.x, puntoC.y) 8. puntoA.x 🡨 puntoC.x 9. puntoA.y 🡨 puntoC.y 10. fin\_mientras 11. fin |

Captura de Processing:



Resultado en Lienzo:



Ejercicio 21: 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.

Analisis:

Datos de Entrada: numero de líneas y circulos

Datos de Salida: círculos y líneas solicitadas mostradas en el lienzo.

Proceso:

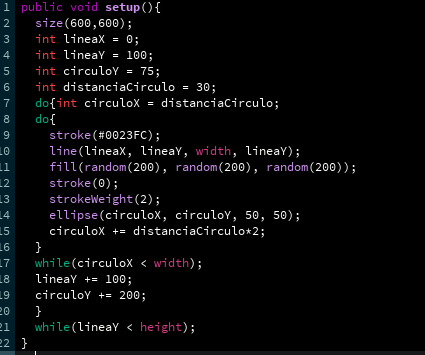
¿Quién Debe realizar el proceso?: el programa

¿Cuál es el proceso que resuelve?: colocar líneas repetidamente hacia abajo y colocar círculos entre cierta distancia.

Diseño:

|  |
| --- |
| Entidad que resuelve el problema: el programa |
| Variables:  - ancholienzo  - altolienzo  - lineaX  - lineaY  - circuloY  - distanciaCirculo  - circuloX |
| Nombre del Algoritmo:círculos\_lineas |
| Proceso del algoritmo:   1. Inicio 2. anchoLienzo🡨600 3. altoLienzo🡨600 4. lineaX 🡨 0 5. lineaY 🡨 100 6. circuloY 🡨 75 7. distanciaCirculo 🡨 30 8. hacer circuloX🡨 distanciaCirculo 9. hacer 10. dibujar línea en (lineaX, lineaY, anchoLienzo, lineaY) 11. dibujar circulo en (circuloX, circuloY, 50, 50) 12. circuloX 🡨 circuloX + distanciaCirculo\*2 13. fin\_hacer 14. mientras (circuloX sea menor que ancholienzo) 15. lineaY 🡨 lineaY +100 16. circuloY 🡨circuloY +200 17. fin\_hacer 18. mientras (lineaY sea menor que altoLienzo) 19. fin |

Captura de Processing:



Resultado en Lienzo:

