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

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

*Ing. Carolina Cecilia Apaza*

*Año*

Trabajo Práctico/Actividad

N°

Apellido y Nombre – LU /

Tarifa, Juan Mateo-

TUV000732

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Indice

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

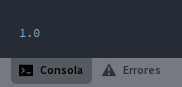
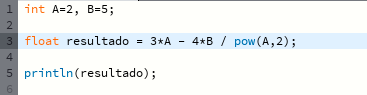
3\* A - 4 \* B / A ^ 2

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

6 –(20/4)

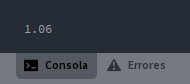
6-5

1

**Captura** **de Processing / Consola**

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

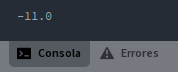
|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | (((4/2)\*3)/6) + ((((6/2)/1)/5^2)/4) \*2 |

**Captura de Processing / Consola**

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.

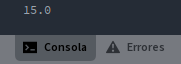
1. b ^ 2 – 4 \* a \* c a= 5, b=3, c=1

|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | b ^ 2 – 4 \* a \* c  (3^2) - ((4\*5)\*1)  9 – 20  -11 |

**Captura de Processing / Consola**

1. 3 \* X ^ 4 – 5 \* X ^ 3 + X 12 – 17 X=2

|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | (3\*(2^4)) – (5\*(2^3)) + (2\*12) – 17  48 – 40 + 24 – 17  15 |

**Captura de Processing / Consola**

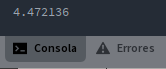
1. (b + d) / (c + 4) b=4, d=5, c=2

|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | (b + d) / (c + 4)  (4 + 5) / (2 + 4)  9 / 6  1.5 |

**Captura de Processing / Consola**

1. (x ^ 2 + y ^ 2) ^ (1 / 2) x=4,y=2

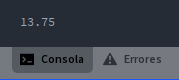
|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | (x ^ 2 + y ^ 2) ^ (1 / 2)  (4^2 + 2^2) ^ (1/2)  20 ^ (1/2)  4.47 |

**Captura de Processing / Consola**

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:

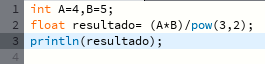
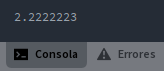
1. B \* A – B ^ 2 / 4 \* C

|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | B \* A – B ^ 2 / 4 \* C  5\*4 – ((5^2) /4) \* C  20 – 6.25  13.75 |

**Captura de Processing / Consola**

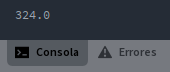
1. (A \* B) / 3 ^ 2

|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | (A \* B) / 3 ^ 2  (4\*5)/3^2  2.22…. |

**Captura de Processing / Consola**

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

|  |  |
| --- | --- |
| ALGEBRAICA | ARITMÉTICA |
|  | (((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 |

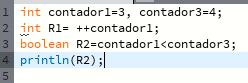
**Captura de Processing / Consola**

Ejercicio 6: Para x=3, y=4; z=1, evaluar el resultado de   
R1 = y+z  
R2 = x >= R1

**Captura de Processing / Consola**

Ejercicio 7: Para contador1=3, contador3=4, evaluar el resultado de   
R1 = ++contador1  
R2 = contador1 < contador2

**Captura de Processing / Consola**



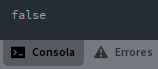
Ejercicio 8: Para a=31, b=-1; x=3, y=2, evaluar el resultado de a+b-1 < x\*y

**Captura de Processing / Consola**



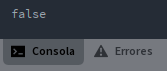
Ejercicio 9: Para x=6, y=8, evaluar el resultado de

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

**Captura de Processing / Consola**

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

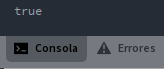
!((i>4) || !(j<=6))

**Captura de Processing / Consola**



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

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

**Captura de Processing / Consola**

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.

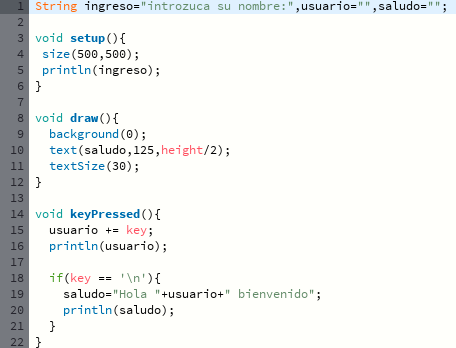
**Análisis**:

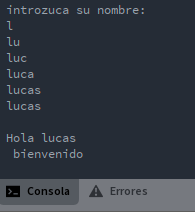
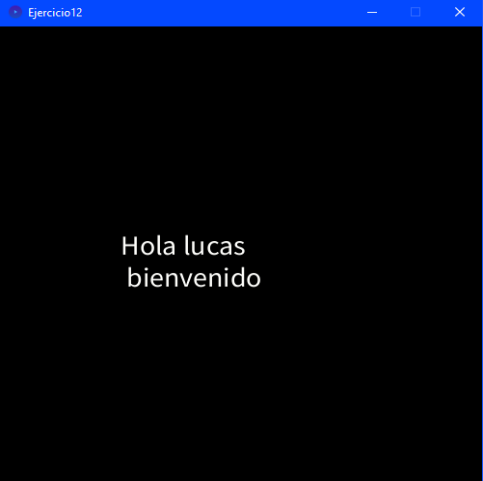
**Datos de Entrada**: nombre   
**Datos de Salida**: saludo en pantalla   
**Proceso**: ¿Quién lo realiza? Algoritmo  
**¿Qué proceso realiza?** Se ingresa el nombre, luego el programa lee ese nombre y devuelve el saludo con el nombre ingresado y se muestra en el lienzo.

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el problema**: Algoritmo |
| **Variables**  **usuario:** string //nombre proporcionado  **saludo:** string //saludo que devuelve la consola |
| **Nombre de algoritmo**: saludo\_usuario |
| * Inicio * *Leer* usuario * saludo”Hola”+usuario+”bienvenido” * *Mostrar* saludo * fin |

**Captura de Processing**



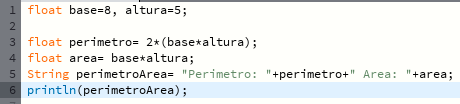
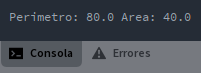
**Consola / Lienzo**

**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  
Datos de Salida: perímetro, área   
Proceso: ¿Quién realiza el proceso? El usuario  
¿Qué proceso realiza? Dados la base y la altura, realiza el cálculo del perímetro y luego del área del rectángulo usando la formulas

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el problema**: Usuario |
| **Variables**  base: float  altura: float  perimetro: float  area: float  perimetroArea: string |
| **Nombre de algoritmo:** resultado\_perimetroArea |
| * inicio * *leer* base * *leer* altura * perimetro2\*(b\*a) * areab\*a * perimetroArea”Perimetro: ”+perímetro+” Area: ”+área * *mostrar*perimetroArea * *fin* |

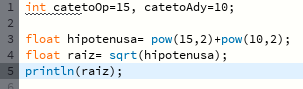
**Captura de Processing / Consola**

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: Cateto adyacente, cateto opuesto  
Datos de salida: Hipotenusa  
Proceso: ¿Quién realiza el proceso? El usuario  
¿Qué proceso realiza? Teniendo los datos de los catetos, se procede a encontrar el resultado de la hipotenusa utilizando la formula.

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el problema**: Usuario |
| **Variables**:  catetoOp: int  catetoAdy: int  hipotenusa: float  raiz: float |
| **Nombre del algoritmo**: calcular\_hipotenusa |
| * inicio * *Leer* catetoOp * *Leer* catetoAdy * *hipotenusa(catetoOp^2 + catetoAdy^2)* * raiz * *mostrar* raiz * *fin* |

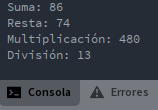
**Captura de Processing / Consola**

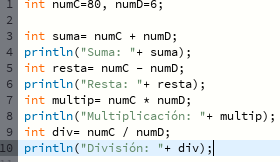
Ejercicio 15: Si viste algo de los apuntes y vídeos, esto debería ser muy fácil de resolver. Dados dos números permita calcular la suma, resta, multiplicación y división de estos. Considere que cada una de estas operaciones es un algoritmo cuando realice el diseño. Obviamente muestre los resultados.

**Análisis**:   
**Datos de entrada**: numC, numD  
**Datos de salida**: suma, resta, multiplicación, división   
**Proceso**: ¿Quién realiza el proceso? Calculadora o usuario  
¿Qué proceso realiza? Identifica los 2 números y luego comienza con las operaciones siguiendo un orden.

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el problema**: Calculadora |
| **Variables**:  numC= int  numD= int  suma= int  resta= int  multip= int div= int |
| **Nombre de algoritmo**: calcular\_operaciones |
| * inicio * *Leer* numC * *Leer* numD * sumanumC + numD * *mostrar* “Suma: “+ suma * restanumC – numD * *mostar* “Resta: “+ resta * multipnumC \* numD * *mostrar* “Multiplicación: “+ multip * divnumC / numD * *mostrar* “Divisón: “+ div * *fin* |

**Captura de Processing / Consola**

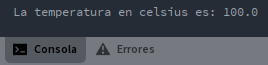


Ejercicio 16**:** Necesitamos convertir una temperatura Fahrenheit en grados Celsius. Si no conoce la forma en la que se realiza esta conversión, debería investigarlo; para eso sirve la etapa de análisis. Pero como somos buenos, daremos una ayuda.

**Análisis:**Datos de entrada: Temperatura en Fahrenheit  
Datos de salida: Temperatura en Celsius  
Proceso: ¿Quién realiza el proceso? La calculadora  
¿Qué proceso realiza? Se toma la temperatura en grados Fahrenheit, luego se la convierte a grados Celsius a través de su fórmula correspondiente

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el problema**: Calculadora |
| **Variables**:  tempFahr: float  tempCel: float |
| **Nombre de algoritmo**: convertir\_temperatura |
| * *inicio* * Leer tempFahr * tempCel(tempFahr – 32) \* (5.0/9.0) * *mostrar* tempCel * *fin* |

**Captura de Processing / Consola**

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.

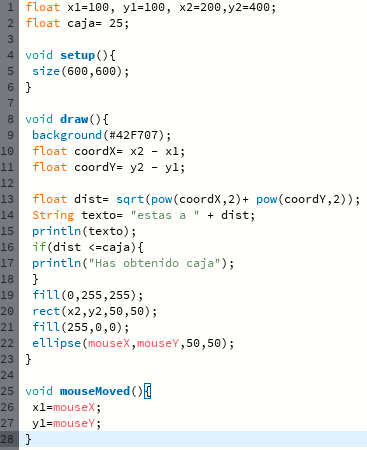
Para esto debe calcular el tamaño de los catetos y luego aplicar el teorema. Halle la distancia entre ambos objetos. Cuando programe, represente a Link con un Circulo, y al tesoro con un cuadrado. Además, mueva a Link mediante el mouse.

**Análisis**:   
Datos de entrada: Coordenadas Link, coordenadas cofre  
Datos de salida: Distancia entre link y cofre  
Proceso: ¿Quién lo realiza? El programa  
¿Qué proceso realiza? Calculamos las diferencias entre las X e Y, obtenemos los catetos para lleva a cabo el teorema de Pitágoras

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el problema**: Programa |
| **Variables**:  x1, y1, x2, y2 = float  caja= float  coordX= float  coordY= float  dist= float |
| Nombre del algoritmo: distancia\_linkCofre |
| * Inicio * *Leer* x1 * *Leer* y1 * *Leer* x2 * *Leer* y2 * caja25 * coordX x2 – x1 * coordY y2 – y1 * dist * *mostrar “estas a “+ dist* * ***si*** (dist <= caja) **entonces** * *mostrar* “Has obtenido caja” * *fin\_si* * *fin* |

**Captura de Processing**



**Consola / Lienzo**

|  |
| --- |
| **Marca la distancia a la que se encuentra Link de la caja, al mover el mouse hacia la caja se muestra el mensaje de la caja obtenida** |

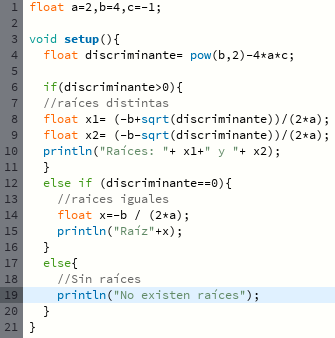
|  |
| --- |
| **Link alcanza el cofre, y aparece el mensaje de la caja** |

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, c  
Datos de salida: Raíces   
Proceso: ¿Quién realiza el proceso? El usuario o calculadora  
¿Qué proceso realiza? Asigna los valores a las letras, luego resuelve el discriminante y luego procede a buscar las raíces

**Diseño**:

|  |
| --- |
| Entidad que resuelve el algoritmo: Usuario |
| Variables:  a= float  b= float  c= float  discriminante= float |
| Nombre del algoritmo: raíces\_ecuacion |
| * *Inicio* * *Leer* a * *Leer* b * *Leer* c * discriminanteb^2 – 4\* a\* c * **si** (discriminante>0) **entonces** * x1 (-b + (discriminante)) / 2\* a * x2 (-b – (discriminante)) / 2\* a * *mostrar* “Raíces: ” + x1 + “ y “+ x2 * **si\_no si** (discriminante==0) **entonces** * x -b / (2\*a) * *mostrar* “Raíz: “ + x * **si\_no** * *mostrar*“No existen raíces” * *fin* |

**Captura de Processing / Consola**

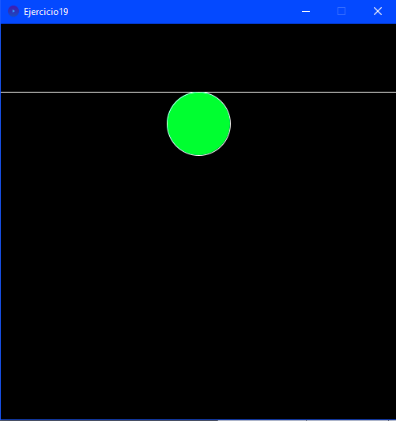
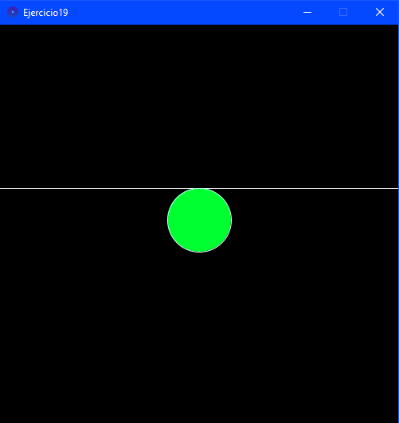
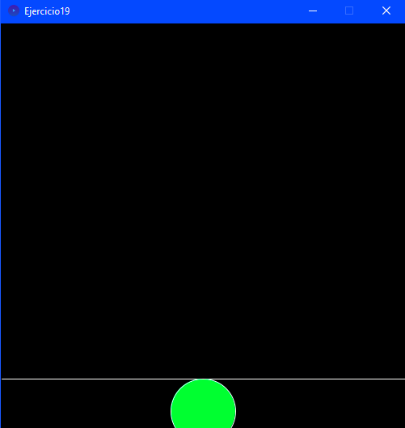
|  |  |  |
| --- | --- | --- |
| 2 RAICES (2,4,-1) | 1 RAIZ(2,4,2) | NINGUNA RAIZ(1,1,1) |
|  |  |  |

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.

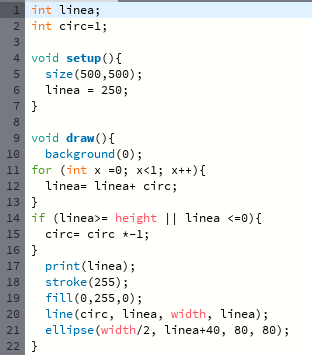
**Análisis**:  
Datos de entrada: distancia del punto, ancho y alto del punto  
Datos de salida: linea y circulo en bucle  
Proceso: ¿Quién realiza el proceso? El programa  
¿Qué proceso realiza? Procesa los datos sobre la línea y el punto, luego grafica ambos de manera constante a través de un bucle

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el algoritmo: Lienzo** |
| **Variables:**  linea= int  circ= int |
| **Nombre del algoritmo: bucle\_lineaCirculo** |
| * Inicio * *Leer*linea * *Leer*circ * *anchoLienzo*🡸500 * altoLienzo🡸500 * **para** x0 **hasta** x<1 **con paso** incremento 1 **hacer** * linealinea + circ * *fin\_para* * **si** linea>=altoLienzo O linea<=0 **entonces** * circcirc\*(-1) * *fin\_si* * *mostrar* linea * *dibujar* linea en (circ, linea, anchoLienzo, lienzo) * *dibujar* criculo en (anchoLienzo/2, linea+40, 80, 80) |

**Lienzo**

**Captura de Processing**



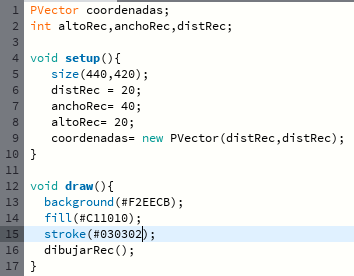
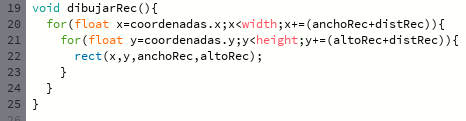
Ejercicio 20**:** Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for. El lienzo debería verse así:

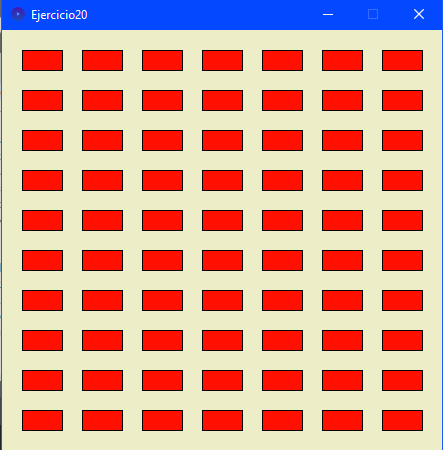
**Análisis:   
Datos de entrada**: El ancho y alto del lienzo.  
Los valores del alto, ancho del rectángulo y distancia entre los rectángulos.

El dibujo de los rectángulos en el lienzo.  
**Datos de salida**: Los rectángulos dibujados en el lienzo  
Proceso: ¿Quién realiza el proceso? Un programa, en este caso, processing  
¿Qué proceso realiza? Dibuja rectángulos en el lienzo con las medidas que se especifican (teniendo en cuenta su ancho, alto y distancia entre cada uno), hacemos uso de un bucle para lograr la tarea.

**Diseño:**

|  |
| --- |
| **Entidad que resuelve el algoritmo: Programa** |
| **Variables:**  anchoRect, altoRect, distEntreRect: entero  altoLienzo, anchoLienzo: entero  coordenadas: float //almacenan coordenadas en x,y |
| **Nombre del algoritmo:** dibujarRec |
| * *Inicio* * anchoLienzo ← 440 * altoLienzo ← 420 * distRec ← 20 * anchoRec ← 40 * altoRec ← 20 * **para** x=coordenadas.x **hasta** anchoLienzo **con paso** (anchoRec+distRect) * **hacer** * **para** y=coordenadas.y **hasta** altoLienzo **con paso** (altoRect+distRect) * **hacer** * *dibujar* rectángulo en (x,y,anchoRec,altoRec) * *fin\_para* * *fin\_para* * *fin* |

**Captura de Processing**

**Lienzo**

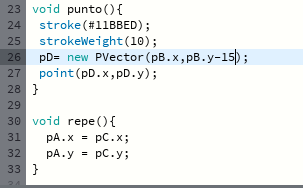
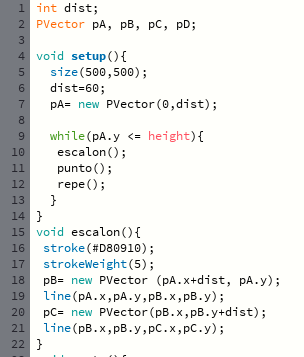
Ejercicio 21**:** Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto de color rojo.

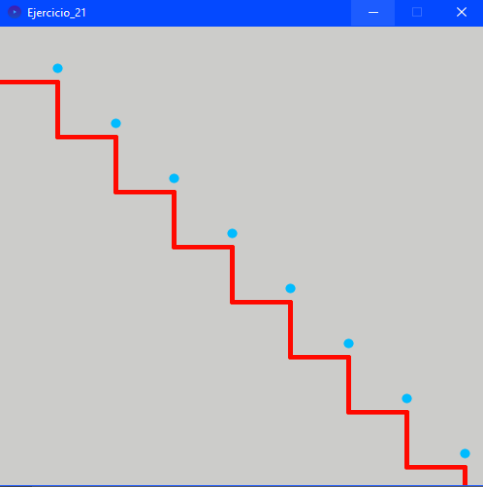
El tamaño del lienzo es size(500,500). La estructura while() se ejecuta dentro de la función setup(). La condición es que solo se dibuje dentro del lienzo. Utilice variables que puedan ayudar a la construcción del dibujo, por ej: x, y, anchoEscalon, altoEscalon, etc.

**Análisis**:   
**Datos de entrada**: puntoA, puntoB, puntoC, puntoD, distancia, alto y ancho del lienzo  
**Datos de salida**: Escalones con puntos en los bordes, mostrado en el lienzo  
**Proceso**: ¿Quién realiza el proceso? El programa  
¿Qué proceso resuelve? Mediante el bucle while() repite el proceso de dibujar escalones y puntos rojos en sus bordes

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el algoritmo**: Programa |
| Variables:  pA, pB, pC, pD: int //vectores  dist= int |
| Nombre del algoritmo: escalones\_puntos |
| * Inicio * anchoLienzo500 * altoLienzo * dist * mientras (pA <= anchoLienzo) **hacer** * *dibujar* linea horizontal en (pA.x, pA.y, pB.x, pB.y) * *dibujar* línea vertical en (B.x, pB.y, pC.x, pC.y) * *dibujar* circulo en (pD.x, pD.y) * pA.x ← pC.x * pA.y ← pC.y * *fin\_mientras* * *fin* |

**Captura de Processing**

**Lienzo**

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

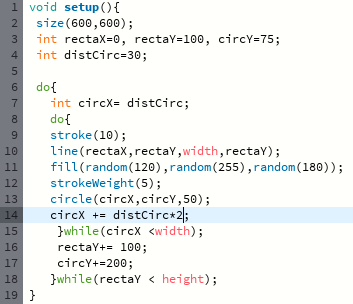
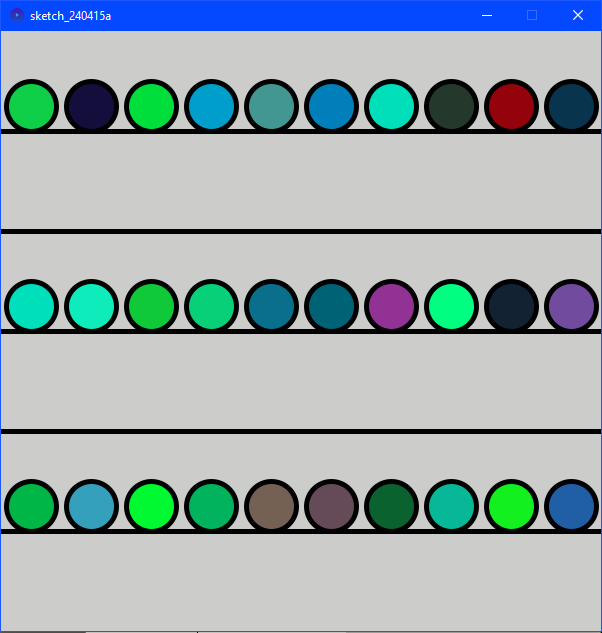


La imagen debe ser construida desde la función setup(). Defina el tamaño del lienzo en size(600,600), verticalmente se divide el lienzo en franjas de igual medida, se deben dibujar los círculos sobre cada línea de por medio es decir en la línea 1 se dibujan círculos con distanciamiento, en la línea 2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

**Análisis**:  
**Datos de entrada**: líneas, círculos, ancho y alto del lienzo  
**Datos de salida**: bolitas de diferente color sobre líneas distanciadas  
**Proceso**: ¿Quién realiza el proceso? Programa  
¿Qué proceso realiza? Divide el lienzo en líneas alternadas, con círculos de diferentes colores esparcidos sobre ellas

**Diseño**:

|  |
| --- |
| **Entidad que resuelve el problema**: Programa |
| **Variables**:  distCirc= int  rectaX, rectaY, circX, circY, distCirc= int  anchoLienzo, altoLienzo: int |
| **Nombre del algoritmo**: circulo\_repeticion |
| * *Inicio* * anchoLienzo ← 600 * altoLienzo ← 600 * rectaX ← 0 * rectaY ← 100 * distCirc ← 30; * circY ← 75 * **hacer** * circX ← distCirc * **hacer** * *dibujar* linea en (rectaX, rectaY, anchoLienzo, rectaY) * *dibujar* circulo en (circX, circY, 50) * circX ← circX + distCirc\*2 * **fin\_hacer** * **mientras** (circuloX < ancholienzo) * rectaY ← rectaY + 100; * circY ← circY + 200; * **fin\_hacer** * **mientras** (lineaY sea menor que altoLienzo) * *fin* |

**Captura de Processing / Lienzo**

Fuentes bibliográficas

Canales de ayuda

<https://www.youtube.com/@noobscourse1591/videos>

<https://www.youtube.com/@DanielMarcial22>

<https://www.youtube.com/@Airroom/videos>

<https://www.youtube.com/@arielvega3350>

Fahrenheit a Celsius (Conversor de Google)

<https://www.google.com/search?client=opera-gx&q=convertir+de+fahrenheit+a+celsius&sourceid=opera&ie=UTF-8&oe=UTF-8>

Tutorial de Processing 3.0 – Buenos Aires Ciudad

Hola Mundo con Processing- Roció Abascal, Erick López, Sergio Zepeda- 2015