Texto

Descripción generada automáticamente con confianza media

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Trabajo Práctico N.º 1

DEPARDO CATRIEL

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

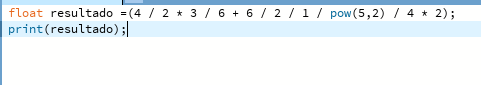
ejercicio 2: Evaluar la siguiente expresión 4 / 2 \* 3 / 6 + 6 / 2 / 1 / 5 ^ 2 / 4 \* 2

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

 1 + 6 / 2 / 1 / 25 / 4 \* 2

1. + 0,06

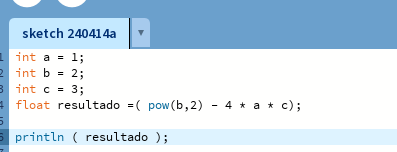
**1,06**



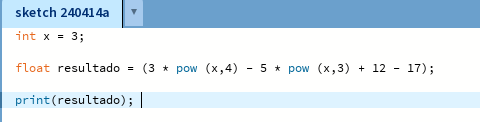
**ejercicio 3:**  Escribir las siguientes expresiones algebraicas como expresiones algorítmicas (en su forma aritmética dentro del algoritmo). En este caso no se pide evaluarlas ni programarlas.

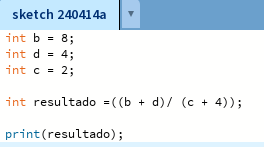
1. ( A ^ 2) ^  ½ - 4 \* A \* C
2. ( X ^  2 + Y ^ 2) / Z ^ 2
3. (3 \* X + 2 \* Y) / 2 \* Z
4. (A + B) / (C - D)
5. 4 \* X ^ 2 - 2 \* X + 7
6. (X+ Y) / X - (3 \* X) / 5
7. A / (B \* C)
8. X \* Y \* Z
9. (A - C) / (B - D)
10. 2 \* 3.14
11. A ^ 2 - B ^ 2 + C ^ 2 - D ^ 2

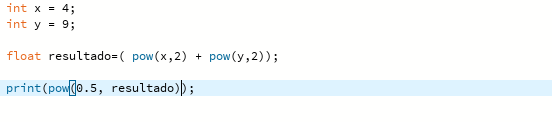
ejercicio 4: Evaluar las siguientes expresiones aritméticas, para lo cual indicar en el caso de las variables, el valor indicado. Luego escribirlas como expresiones algebraica











**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:

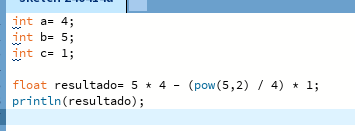
a)B \* A – B ^ 2 / 4 \* C

5 \* 4 – 5 ^ 2 / 4 \* 1

20 – 25 / 4 \* 1

20 – 6,25

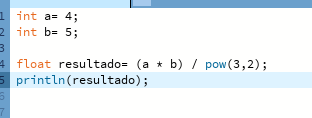
**13,75**



**b) (**a \* b) / 3 ^ 2

4 \*5 / 3 ^ 2

20 / 9 = 2,22



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

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

((6 / 2 \* 4 + 10) \* 3 \* 5) – 6

((6 / 2 \* 4 + 10) \* 3 \* 5) – 6

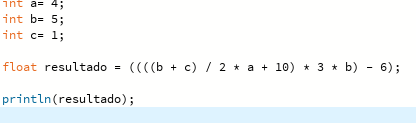
((3 \* 4 + 10) \* 3 \* 5) – 6

((12 + 10) \* 15) – 6

(22 \* 15) – 6

330 – 6

**324**



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

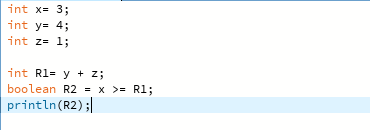
R1= y + z

R2= x > = R1

R1= 4 + 1 = 5

R2= 3 > R1

FALSE

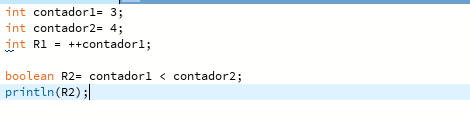


Ejercicio 7: Para contador1=3, contador2=4, evaluar elresultado de R1 = ++contador1 R2 = contador1 < contador2

R1= ++3= 4

R2= 4<4

FALSE



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

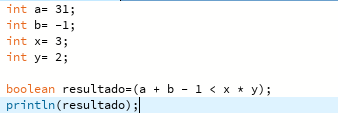
a + b – 1 < x \* y

31 – 1 – 1 < 3\* 2

31 – 1 – 1 < 6

29 < 6

FALSE



Ejercicio 9: Para x=6, y=8, evaluar elresultado de !(x=7)

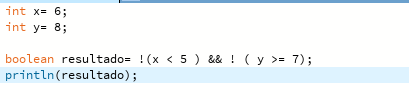
!(X < 5) cc !(y >=7)

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

!(falso) cc ¡(verdadero)

Verdadero cc falso

Falso



Ejercicio 10: Para i=22,j=3, evaluar el resultado de !((i>4) || !(j<=6))

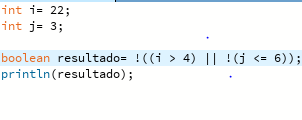
!((i > 4 ) ||!(j <= 6))

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

¡(true || false)

¡(true)

FALSE



Ejercicio 11: 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) CC (b – c >= 19)

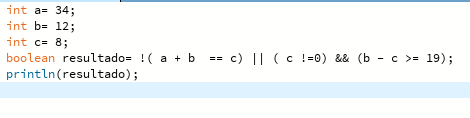
!(34 + 12 == 8) || ( 8 != 0 ) CC (12 – 8 >= 19)

¡(46 == 8) || ( true) CC (4 >= 19)

¡(false) || false

True || false

True



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.

**ANALISIS:**

Datos de entrada: ingresamos el nombre a travez del teclado

Proceso: se determinan las teclas pulsadas por el usario

Datos de salida: se saluda en la pantalla con el nombre que anteriormente habíamos ingresado en el teclado

**DISEÑO:**

String consola: almacena el texto que se muestra en la consola de abajo

String nombre: guarda el nombre ingresado

String saludo: almacena el saludo que podemos personalizar y se muestra en la pantalla

Funciones:

En la función setup se configura el tamaño del lienzo y se muestra el texto en la pantalla

En la función draw se dibuja el saludo

En la función keyPressed se quedan guardadas las teclas presionadas

Generación del saludo:

If : se verifica si el usario apretó la tecla “enter”

Si la tecla presionada es “enter” se generara el saludo con el nombre que se haya escrito

La estructura del saludo es:

Saludo= “!hola “ + “nombre “ + “¡”



Ejercicio 13: 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 del rectángulo y altura del rectángulo

Proceso: se calcula el perímetro del rectángulo utilizando la formula:2 \*(base por altura).se calcula el área del rectángulo utilizando la formula base \* altura

Datos de salida: perímetro y area del rectángulo

**DISEÑO:**

Se calcula el área del rectángulo multiplicando la suma de la base y la altura por 2 y se almacena en la variable perímetro

Calculo de área:

Se calcula el área del rectángulo multiplicando la base por la altura y se almacena en la variable del área

Mensaje:

String pa= “el perímetro es de:” + “perímetro” + “y el área es de:” + área + “ perteneciente al rectángulo”;

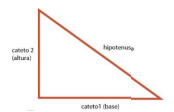
Se generara un mensaje uniendo las variables del perímetro y área junto con el texto descriptivo. El mensaje se almacena en la variable pa.

Presentacion del mensaje:

Se imprime el mensaje almacenado en la variable pa en la consola utilizando la función println(pa);



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



**ANALISIS:**

Datos de Entrada: base del cateto1 y altura del cateto2

Proceso: se calcula la hipotenusa del triangulo rectángulo usando el teorema de Pitágoras(hipotenusa ˄2 = (cateto1˄2 + cateto2˄2)

Datos de Salida: longitud de la hipotenusa

**DISEÑO:**

Calculo de la hipotenusa:

Pow(pow(cateto1,2) + pow(cateto2,2),0,5);

Se calcula la hipotenusa usando funciones pow para asi elevar al cuadrado los dos catetos,luego sumamos el resultado de ambos y le sacamos su razi cuadrada,dando asi con la longitud de la hipotenusa

Resultado:

Println(hipotenusa);

se dibuja la longitud de la hipotenusa en la consola usando la función println

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

**ANALISIS:**

Datos de entrada: numero1 y numero2

Proceso: se suman numero1 y numero2,se resta numero2 del numero1 y luego se multiplican numero1 y numero2, para después dividir numero1 por numero2

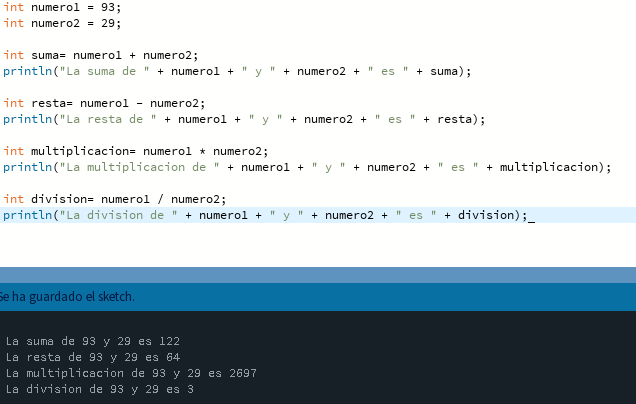
Datos de salida: resultado de la suma de numero1 y numero2,resultado de la resta de numero1 y numero2,resultado de la multiplicación de numero1 y numero2,resultado de la división de numero1 por numero2

**DISEÑO:**

Operaciones aritméticas: usando los valores de numero1 y numero2 se realizan la correspondiente suma,resta,multiplicación y división

Los resultados se almacenan en las variables, suma, resta, multiplicación y división

Presentación del resultado: se muestra en la consola un mensaje para cada operación,mostrando los valores de numero1 y numero2 junto con el resultado de las operaciones



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

**ANALISIS**:

Datos de Entrada: Temperatura en grados Fahrenheit.

Proceso: Consiste en convertir la temperatura del sistema Fahrenheit al sistema de grados Celsius utilizando la fórmula: "temperaturaCelsius = (temperaturaFahrenheit - 32) / 1.8".

Datos de Salida: Temperatura en grados Celsius.

**DISEÑO:**

Conversión:

Se establece el valor de temperaturaFahrenheit en 90, que representa la temperatura en grados Fahrenheit.

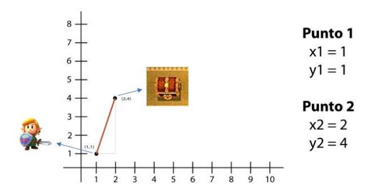
Se aplica la fórmula de conversión de Fahrenheit a Celsius: (temperaturaFahrenheit - 32) / 1.8.

El resultado se guarda en la variable temperaturaCelsius.

Presentación del Resultado:

Se muestra en la consola el valor de temperaturaCelsius, que indica la temperatura convertida a grados Celsius.



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

**ANALISIS:**

Datos de entrada: Coordenadas de Link: (x1, y1). Coordenadas del cofre: (x2, y2). Dimensiones del círculo y cuadrado: 50 x 50 píxeles.

Proceso: Se definen las coordenadas de Link y del cofre. Se calcula la distancia entre Link y el cofre. Si la distancia entre el círculo y el cuadrado es menor que el valor específico de 49 píxeles, el cuadrado no se dibuja, lo que indica que el círculo ha llegado al cuadrado. Si la distancia es mayor o igual a 49 píxeles, el cuadrado se dibuja en su posición.

Datos de salida: Visualización en pantalla del círculo "Link" y el cuadrado "cofre".

**DISEÑO:**

Variables:

linkx = x1: Coordenada x del círculo "Link".

linky = y1: Coordenada y del círculo "Link".

cofrex = x2: Coordenada x del cuadrado "cofre".

cofrey = y2: Coordenada y del cuadrado "cofre".

catetox = (cofrex - linkx): Valor del cateto X.

catetoy = (cofrey - linky): Valor del cateto Y.

Proceso de las coordenadas:

Se establecen las coordenadas iniciales del círculo "link" (x1, y1) y del cuadrado "cofre" (x2, y2).

En la función setup(), se dibuja el círculo "link" y el cuadrado "cofre" en sus coordenadas.

Actualizar posición del círculo:

En la función draw(), se actualizan continuamente las coordenadas del círculo "link" con la posición del ratón.

void draw() {

linkx = mouseX

linky = mouseY

}

Calcular distancia:

Se calcula la distancia entre el círculo "link" y el cuadrado "cofre" en cada iteración de draw().

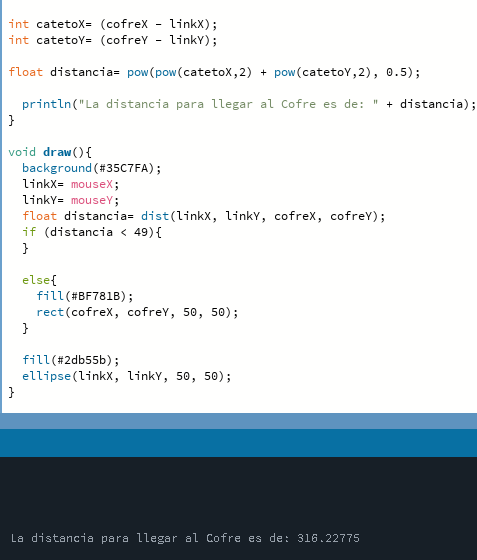
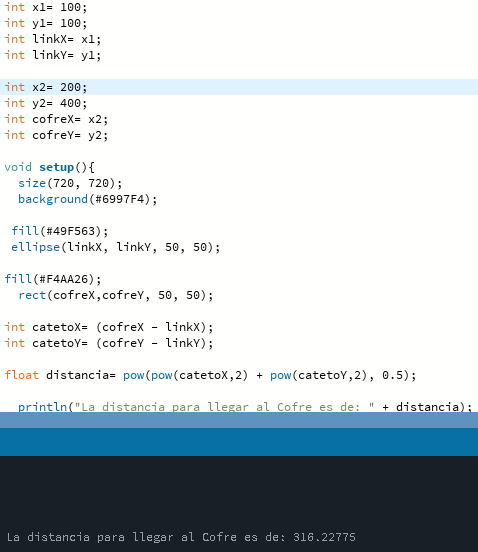
Llegada al Cofre:

Se comprueba si la distancia es menor que 49 píxeles.

Si la distancia es menor, el cuadrado "cofre" no se dibuja, indicando que el círculo "link" ha llegado a su posición objetivo.

float distancia = dist(linkx, linky, cofrex, cofrey);

if (distancia < 49) { }



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.

**ANALISIS:**

Entrada de datos: a, b, c son los coeficientes de la ecuación cuadrática.

Proceso:

Calcular el discriminante utilizando la fórmula: (b ^ 2) - 4 \* a \* c.

Salida de datos: Se mostrarán las raíces de la ecuación cuadrática (si existen) o un mensaje indicando que no hay raíces reales.

**DISEÑO:**

Variables:

a, b, c: Coeficientes de la ecuación cuadrática.

discriminante: Valor del discriminante de la ecuación.

x1, x2: Raíces de la ecuación cuadrática.

Proceso del código:

Calcular el discriminante utilizando la fórmula: (b ^ 2) - 4 \* a \* c.

Verificar el valor del discriminante:

Si el discriminante es mayor que cero, calcular las dos raíces reales utilizando la fórmula:

x = (-b ± √(discriminante))/ (2 \* a)

if (discriminante > 0){

float x1 = (-b + sqrt(discriminante)) / (2 \* a);

float x2 = (-b - sqrt(discriminante)) / (2 \* a);

println("Las raíces son: " + x1 + " y " + x2);

}

Si el discriminante es igual a cero, calcular la raíz doble utilizando la fórmula: x = -b / (2 \* a)

else if (discriminante == 0){

float x = -b / (2 \* a);

println("La raíz doble es: " + x)

}

Si el discriminante es menor que cero, no existen raíces reales.

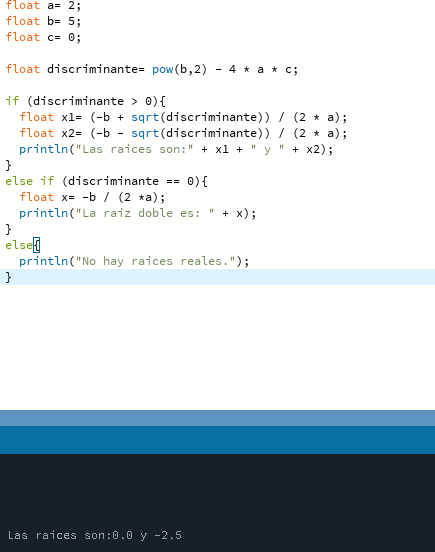
else{

println("No hay raíces reales.");

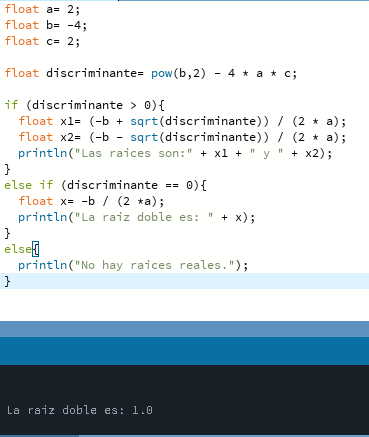
}

Mostrar las raíces calculadas (si existen) o un mensaje indicando la ausencia de raíces reales.

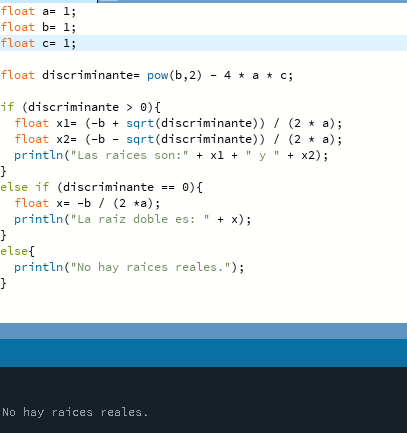
A=2 , B=5 , C=0



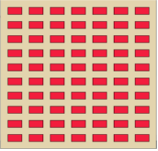
A=2 , B=-4 , C=2



A=1 , B=1 , C=1



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:



**ANALISIS:**

Entrada de datos: Dimensiones del lienzo, ancho de 440 píxeles y altura de 420 píxeles. Dimensiones de los rectángulos, ancho de 40 píxeles y altura de 20 píxeles. Separación entre los rectángulos, 20 píxeles.

Proceso: Se calcularán las coordenadas X y Y de cada rectángulo desde su posición en la cuadrícula y la separación entre ellos. Se trazarán los rectángulos en las posiciones calculadas.

Datos de salida: El lienzo con la cuadrícula de rectángulos trazados en él. Cada rectángulo estará en una posición específica en el lienzo, con una separación de 20 píxeles tanto horizontal como verticalmente.

**DISEÑO:**  
  
Configuración inicial: Establecí el tamaño de la ventana de visualización en 440 píxeles de ancho y 420 píxeles de alto. Definí un color de fondo amarillo para la ventana y un color para los rectángulos mostrados posteriormente dentro de la ventana.

Definición de variables:

rectanguloAncho: Define el ancho de cada rectángulo como 40 píxeles.

rectanguloAlto: Define la altura de cada rectángulo como 20 píxeles.

distanciaX: Establece la separación horizontal entre los rectángulos como 20 píxeles.

distanciaY: Establece la separación vertical entre los rectángulos como 20 píxeles.

Dibujo de la cuadrícula: Utilizar dos bucles "for" anidados para iterar sobre las posiciones en la ventana. El bucle exterior controla la posición horizontal y el bucle interior controla la posición vertical. Dentro del bucle interior, trazar un rectángulo en cada posición calculada utilizando la función "rect"().

for (int x = centroX; x < width - rectanguloAncho; x += rectanguloAncho + distanciaX) {

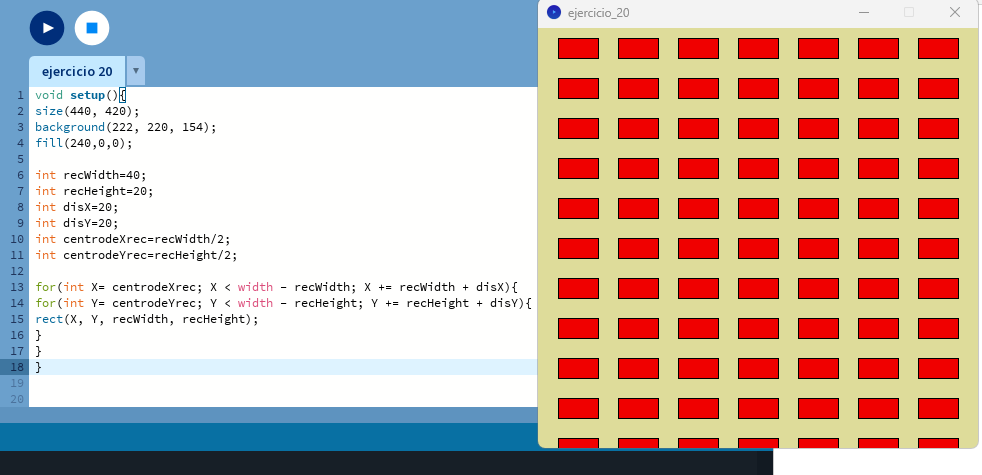
for (int y = centroY; y < height - rectanguloAlto; y += rectanguloAlto + distanciaY) {

rect(x, y, rectanguloAncho, rectanguloAlto);

}

}

Resultado: Al ejecutar el código, se mostrará una ventana de visualización con una cuadrícula de rectángulos trazada en ella. Cada rectángulo tiene un ancho de 40 píxeles, una altura de 20 píxeles y mantiene una separación de 20 píxeles tanto horizontal como verticalmente entre ellos.



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.

**ANALISIS:**  
Entrada de datos: puntoA, puntoB, puntoC, puntoD, son los vectores que representan las coordenadas de los puntos en el escalón.

Proceso: implica repetir mediante while() para dibujar escalones y puntos rojos en los bordes.

Datos de salida: Representación visual de los escalones con puntos rojos en los bordes.

**DISEÑO:**

Variables:

distancia: Representa la separación horizontal entre los puntos en el escalón.

puntoA: Representa la coordenada del primer punto en el escalón.

puntoB: Representa la coordenada del segundo punto en el escalón.

puntoC: Representa la coordenada del tercer punto en el escalón.

puntoD: Representa la coordenada del cuarto punto en el escalón.

Configuración de la ventana: Se establece el tamaño del lienzo como 500 x 500 píxeles y se define el color de fondo.

Proceso del código:

Se asigna un valor a la variable distancia y se crea el primer punto.

Bucle while: Se repite el proceso de dibujo del escalón y el punto rojo mientras puntoA.y sea menor o igual a la altura del lienzo.

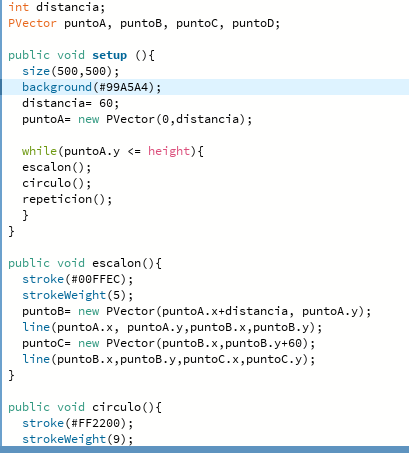
Función escalón(): Dibuja dos líneas horizontales para formar un escalón a partir de puntoA.

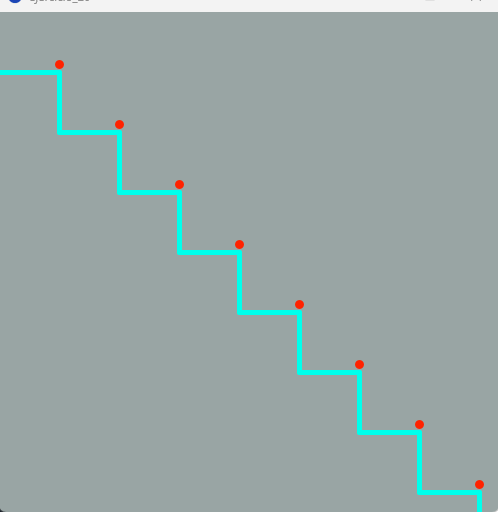
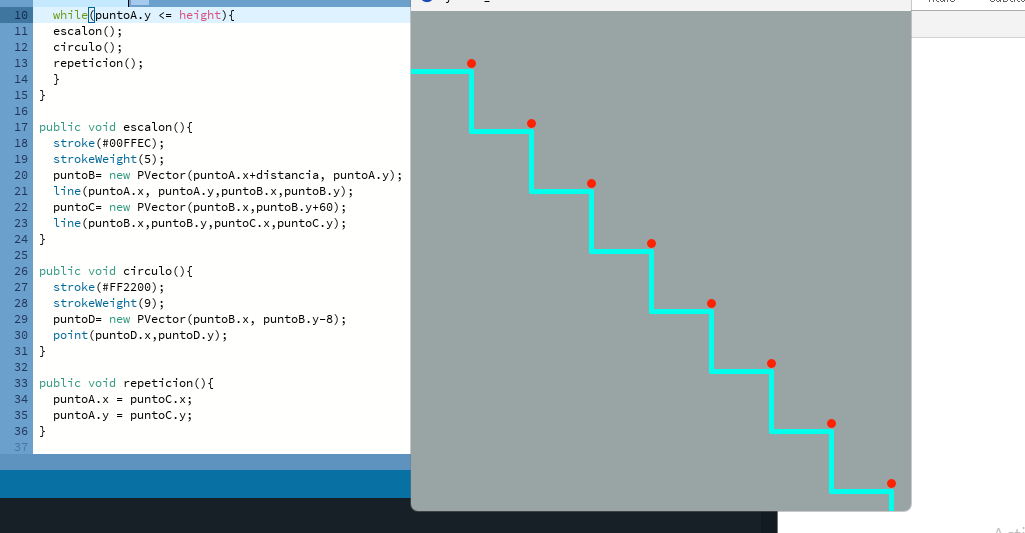
Función círculo(): Dibuja un punto rojo en la esquina superior derecha del escalón a partir de puntoB.

Actualización de coordenadas: La función repetición() actualiza las coordenadas de puntoA para el siguiente escalón.

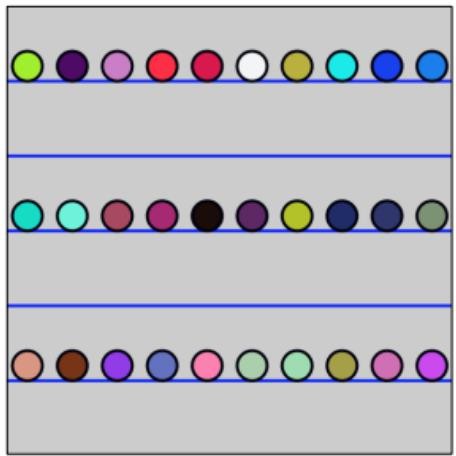
Resultado:

Los escalones se forman de manera descendente desde la esquina superior izquierda del lienzo y se desplazan hacia la derecha con una separación horizontal determinada por la variable distancia. El proceso continúa hasta que se alcance la altura máxima del lienzo. El resultado visual es una estructura escalonada con puntos rojos en cada esquina superior derecha.





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.

**ANALIS:**

Datos de entrada: Dimensión del lienzo: 600x600 píxeles. Franjas verticales de tamaño uniforme. Coordenadas y espacios para las líneas y círculos.

Proceso: El lienzo se divide verticalmente en franjas de igual tamaño. Se trazan líneas horizontales en cada franja. Se alternan dibujos de círculos con colores al azar y espacios uniformes entre ellos.

Datos de salida: Una imagen con franjas verticales que contienen líneas horizontales y círculos intercalados, cada uno con un color aleatorio. Los círculos están distribuidos uniformemente a lo largo de las líneas.

**DISEÑO:**

Variables:

posicionXLinea: Coordenada inicial en el eje X para las líneas.

posicionYLinea: Coordenada inicial en el eje Y para las líneas.

posicionYCirculo: Coordenada inicial en el eje Y para los círculos.

espacioEntreCirculos: Espacio entre los círculos a lo largo de una línea.

Proceso del algoritmo:

Inicialización del lienzo con tamaño 600x600.

Definición de variables para controlar las coordenadas y espacios.

Bucle externo (do-while) para iterar sobre las franjas verticales.

Bucle interno (do-while) para trazar líneas y círculos en cada franja.

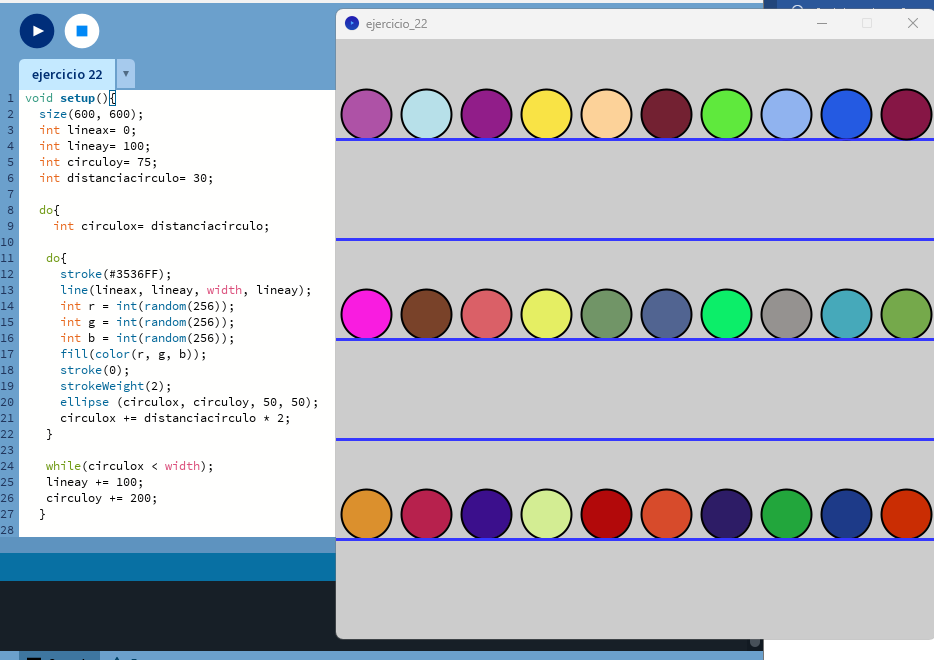
Alternancia entre líneas y círculos.

Generación de colores aleatorios para los círculos.

Avance vertical a la siguiente franja.

Resultado:

El lienzo se divide verticalmente en franjas de igual tamaño. En cada franja, se dibujan líneas horizontales alternadas con círculos dispuestos uniformemente. Los círculos tienen colores aleatorios. Este patrón se repite hasta cubrir todo el lienzo verticalmente.



Me apoye y me base en las clases virtuales subidas al aula virtual,también como a los videos compartidos por la misma,aparte de mis propios apuntes para los primeros puntos desarrollados del trabajo practico

